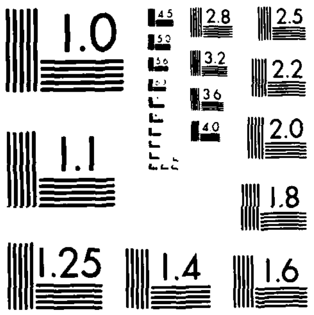
AD-A172 215 SINGER AI PROJECT: KNOWLEDGE-BASED VISION ALGORITHM DEVELOPMENT(U) ENVIRONMENTAL RESEARCH INST OF MICHIGAN ANN ARBOR B T MITCHELL ET AL. AUG 86 ERIM-321700-2-F 1/1

UNCLASSIFIED F/G 9/2 NL

END
DATE
FILMED
11-86



CROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

321700-2-F

AD-A172 215

DTIC FILE COPY

Final Technical Report
Singer AI Project

KNOWLEDGE-BASED VISION ALGORITHM DEVELOPMENT

BRIAN T. MITCHELL, ROBERT C. VOGT,
and JOHN D. OVERMARS

AUGUST 1986

Prepared for:
The Singer Company
Kearfott Division
1150 McBride
Little Falls, New Jersey 07424

10NCK27938

ENVIRONMENTAL
RESEARCH INSTITUTE OF MICHIGAN
BOX 8618 • ANN ARBOR • MICHIGAN 48107

86 9 2

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No. ADA172215	3. Recipient's Catalog No.
4. Title and Subtitle Singer AI Project: Knowledge-Based Vision Algorithm Development		5. Report Date August 1986
		6. Performing Organization Code
7. Author(s) Brian T. Mitchell, Robert C. Vogt, John D. Overmars		8. Performing Organization Report No. 321700-2-F
9. Performing Organization Name and Address Environmental Research Institute of Michigan P.O. Box 8618 Ann Arbor, Michigan 48107-8618		10. Work Unit No.
		11. Contract or Grant No. 10NCK27938
12. Sponsoring Agency Name and Address The Singer Company Kearfott Division 1150 McBride Little Falls, New Jersey 07424		13. Type of Report and Period Covered Final Report Jan. 1986-Aug. 1986
		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

An investigation of applications of Artificial Intelligence (AI) concepts to the process of developing image processing algorithms for Automatic Target Recognition (ATR).

17. Key Words Vision algorithm development Expert system ATR Knowledge acquisition Inference engine	18. Distribution Statement This document has been approved for public release and sale; its distribution is unlimited.

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page)	21. No. of Pages ii + 53	22. Price

CONTENTS

1 INTRODUCTION

This document describes the work performed under the contract titled "Singer AI Project: Knowledge-Based Vision Algorithm Development." It is intended to fulfill the requirements of the final technical report.

1.1 Executive Summary

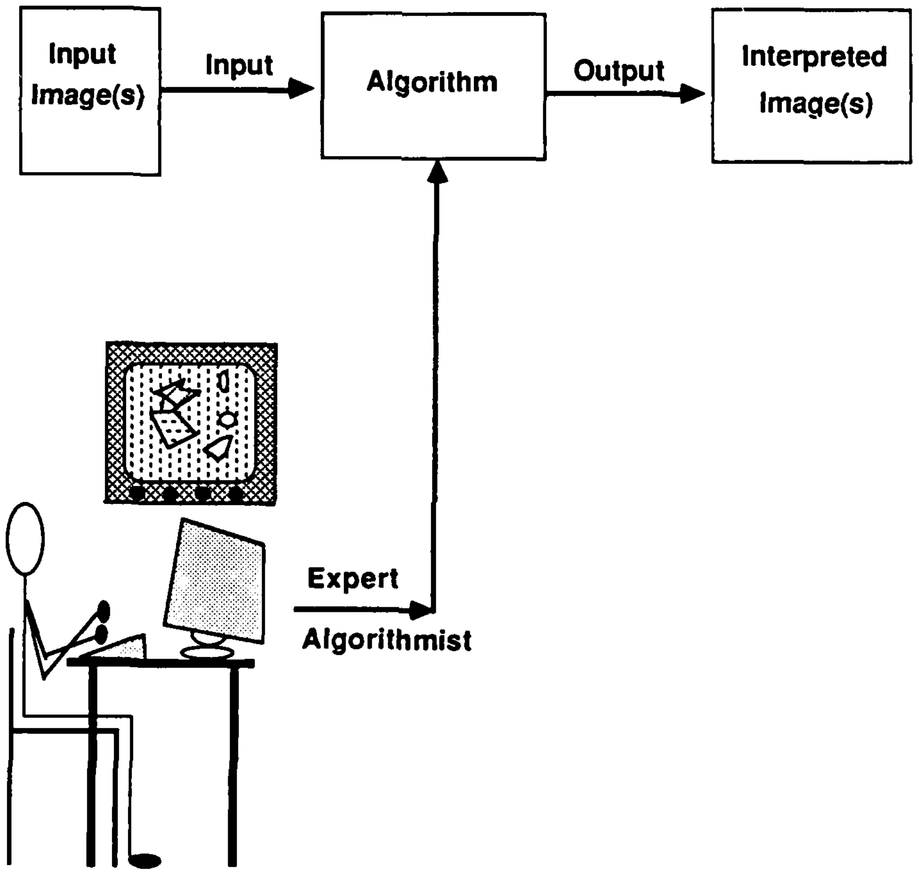
This project was initiated to investigate possible application of Artificial Intelligence (AI) concepts to the process of developing image processing algorithms for Automatic Target Recognition (ATR). Towards this end, a prototype expert system image algorithmist decision aid was designed and implemented. All of the project goals were either met or exceeded. Additional funding is need to fully implement the system defined in this project and integrate it into ERIM's Cyto HSS.

1.2 Motivation

The motivation for this project was based on three major factors. First, the development of machine vision algorithms is the current bottleneck in the integration of vision technologies into application areas. Second, the process of developing such algorithms is currently viewed as an art practiced by only a handful of well-seasoned machine vision experts. Third, and last, Artificial Intelligence techniques have been successfully applied in similar situations in other areas. [illegible]

The process of developing image algorithms is seen as the bottleneck in producing automated computer vision systems. Current approaches are very labor intensive. Routine algorithm development can take several months of dedicated work by highly skilled experts. Thus, the process is both slow and costly. Since such experts are also rare and take up to ten years to develop, the problem is further constrained by the number of current experts.

The process of developing image algorithms is currently viewed as an art. Figure 1-1 (a) depicts this situation. The figure shows that the expert develops the algorithm that takes the input image(s) as input and produces the interpreted image as output, but it does not provide any details as to how that happens. Unfortunately, this captures much of the current understanding of the process.



Current State of Image Algorithm Development

Figure 1-1 (a)

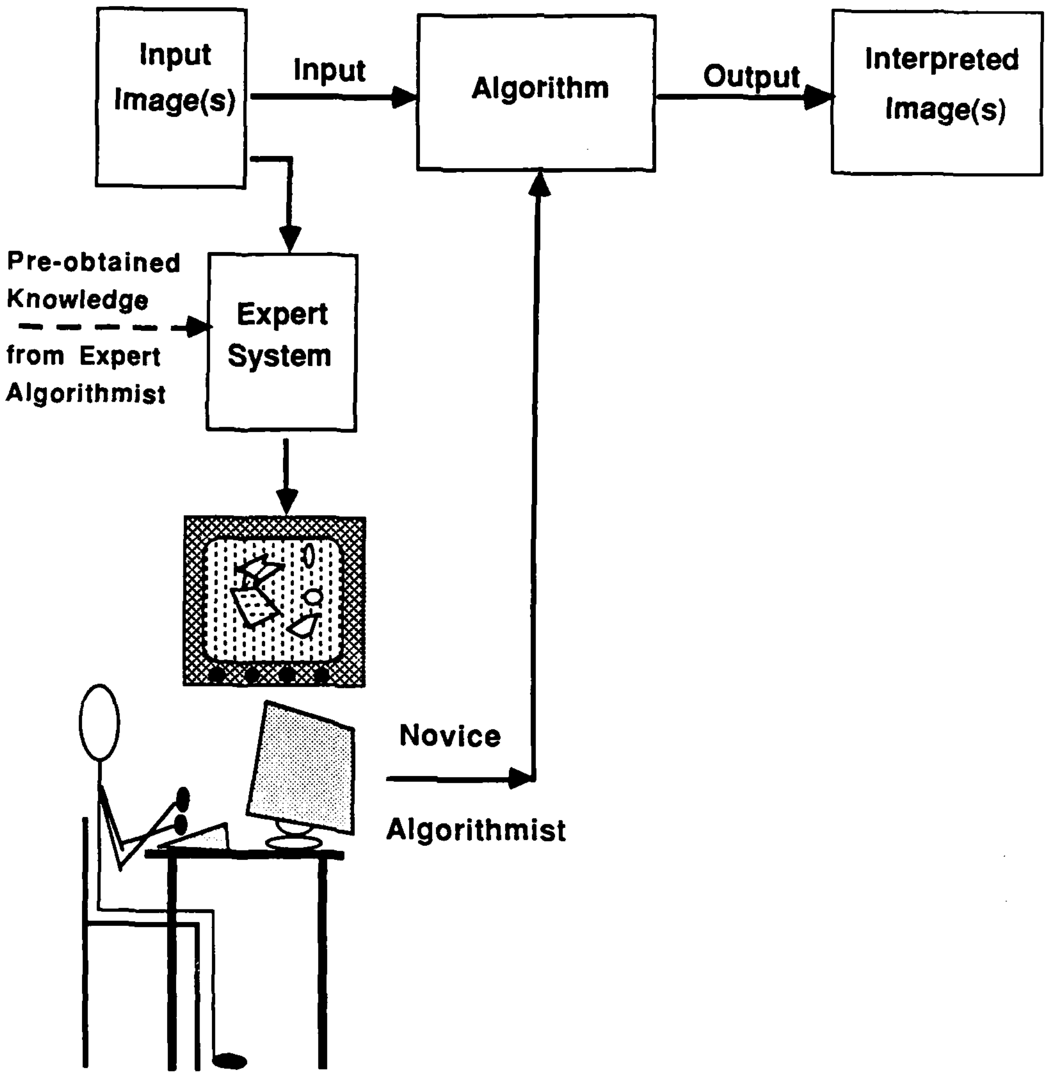
Recent technological developments in applied Artificial Intelligence (expert systems) have successfully provided solutions to problems that have been previously viewed as too complicated. Characteristic of these problems are situations in which things break, information is missing, assumptions fail, and/or mathematical intractability exists. Problems with such characteristics cannot be solved through pat solutions or mathematical models. Instead, expert systems employ heuristic techniques patterned after human reasoning strategies to solve these problems. These techniques have been used to produce systems that perform at expert level in areas such as medical diagnosis, computer configuration, and mineral exploration.

1.3 Project Goals

The project was initiated to address two goals. The first, more near-term goal, was for Singer and ERIM to cooperatively increase their understanding of Artificial Intelligence applications to computer vision. The second, more long-term goal, was to achieve means for much more quickly and economically adapting ATR systems to new target types, scenarios, and other conditions.

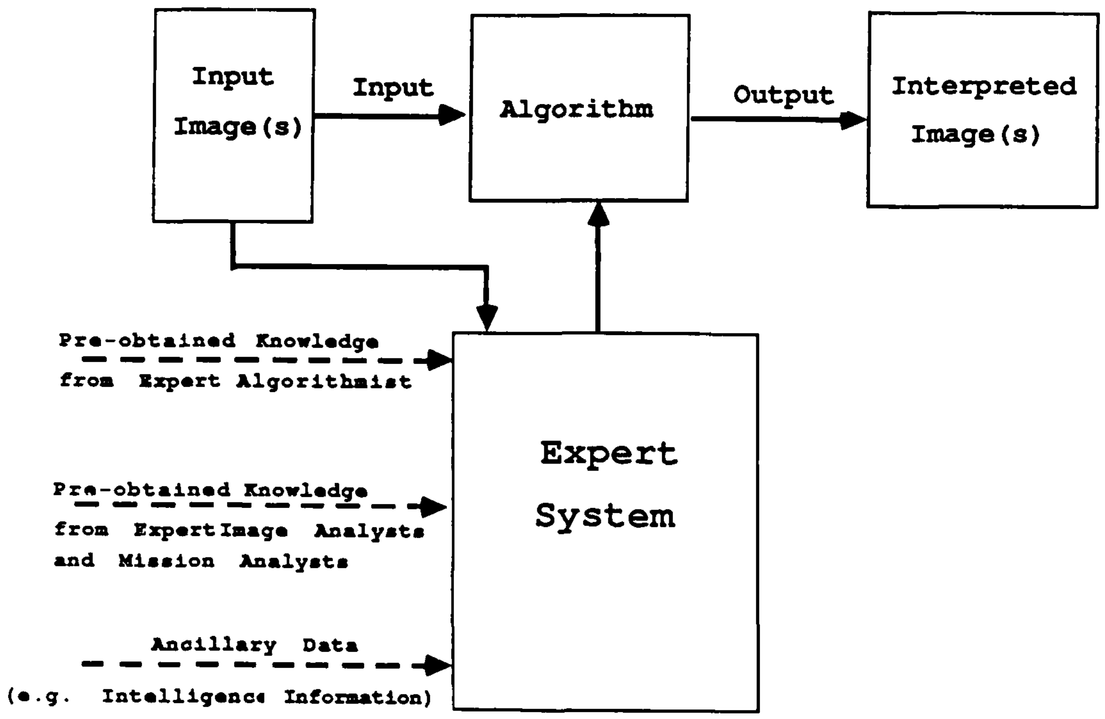
The near-term goals was designed to initiate new ideas and solutions to the problem of developing image processing algorithms. Since both Singer and ERIM have common interest in developing new solutions in this area and since applications of AI ideas to this area are relatively unexplored, it was decided that a combined effort involving both users and researchers was both necessary and worthwhile.

The long-term goal of developing more economical ways to produce ATR image algorithms was viewed as both technologically feasible and essential. Advances in Artificial Intelligence had made progress in this area possible. Integrating this technology into an environment for ATR algorithm development was the challenge. Figure 1-1 (b) illustrates how this was to be accomplished. This figure shows an expert system supporting the development process. Within this expert system is knowledge about how to develop ATR algorithms. This knowledge has been obtained from experts with several years of experience in developing ATR algorithms. The expert system, as illustrated in the figure, makes this knowledge available for the novice user to draw upon, and thus, enables him to developed more sophisticated more robust algorithms. Figure 1-1 (c) illustrates advanced stages of the development of the expert system. It shows an ideal situation where the algorithmist has been fully replaced by an even smarter expert system, and the time required for algorithm development has become a minor factor. Thus, figures 1-1 (a)-(c) show a planned



Desired State of Image Algorithm Development

Figure 1-1 (b)


Input Image(s)
Input
Algorithm
Output
Interpreted Image(s)
Expert System
Pre-obtained Knowledge from Expert Algorithmist
Pre-obtained Knowledge from Expert Image Analysts and Mission Analysts
Ancillary Data (e.g. Intelligence Information)

Ultimate Expert ATR System

Figure 1-1 (c)

sequence of ATR algorithm development support by Artificial Intelligence, from no support to ultimate support. The benefits of following such a plan are numerous and are outlined in table 1-1.

1.4 Project Outcomes

Several very important steps towards ultimate, AI-supported, ATR algorithm development were taken in this project. Table 1-2 summarizes these steps. As shown in the table, this project has resulted in a much increased understanding of the ATR algorithm development process, a vocabulary in which to discuss algorithm concepts, and a collection of global strategies currently used to develop ATR algorithms. In turn, the resulting knowledge lead to the design of an AI-based decision aid for augmenting the algorithm development process. This design provides the detailed specifications for the system outlined in figure 1-1 (b). Two major system components from this design were fully implemented under this project, a knowledge acquisition environment and a inference engine. These components were then used to construct a prototype implementation of the system outlined in figure 1-1 (b).

1.5 Future Directions

Although this project was very successful, there is much work that remains before a fully developed image algorithmist aid is ready for production use. The direction for this work, however, is very clear. Continued knowledge base development is required to implement the remaining phases of the edge extraction algorithm sequence outlined above. The resulting expert system must then be integrated into an AI workstation coupled with ERIM's Cyto HSS.

2 PROJECT PHASES

This project can be broken down into four sequential phases: the problem analysis phase, the system definition phase, the tool development phase, and the knowledge base development phase. In the problem analysis phase, a study was undertaken to explore the essence of image algorithm development. From this phase the project transitioned into a system definition phase in which specifications for a image algorithmist decision aid were developed. These specifications were then followed in the third project phase, the tool development phase. In this phase an inference engine and a visual inference network editor were developed. These tools were then used in phase four to construct a prototype expert system. An overview of the entire project in presented in figure 2-1.

- Broader Availability
- Shorter Development Cycle
- More Robust ATR Decision Algorithms
- Quicker Adaption To New Tactics, Threats, Environments, Countermeasures, etc...
- Greater Ease-of-Use In Incorporating Pre-brief Data Into ATR

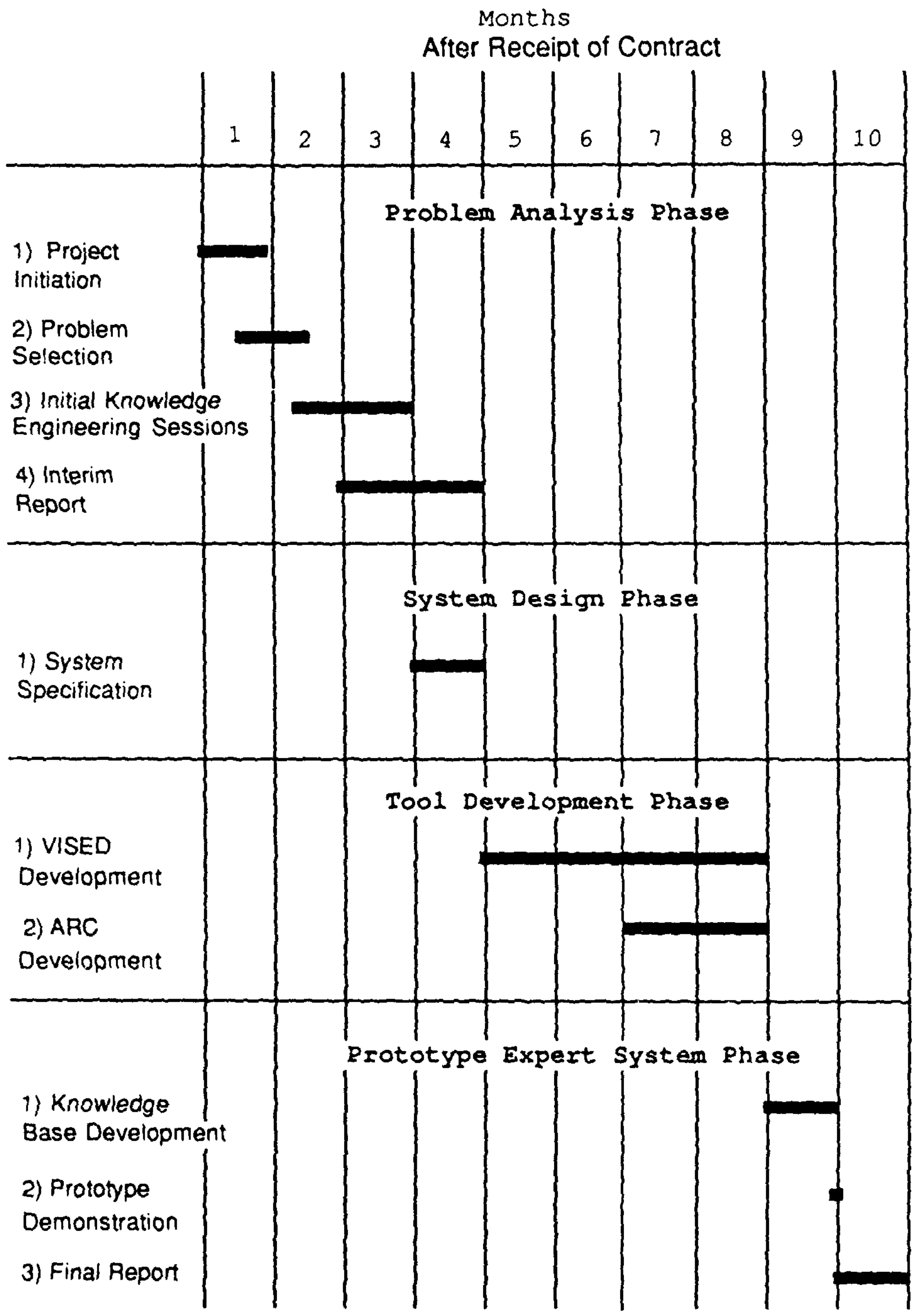
Benefits Achievable From Employing Expert System Technology in ATR's

Table 1-1

- Increased Understanding of Image Algorithm Development
- Vocabulary of Image Processing Terms
- Global Strategies
- System Design
- Knowledge Acquistion Environment
- Inference Engine
- Prototype Expert System

Project Outcomes

Table 1-2



Project Overview

Figure 2-1

3 SOFTWARE SYSTEMS

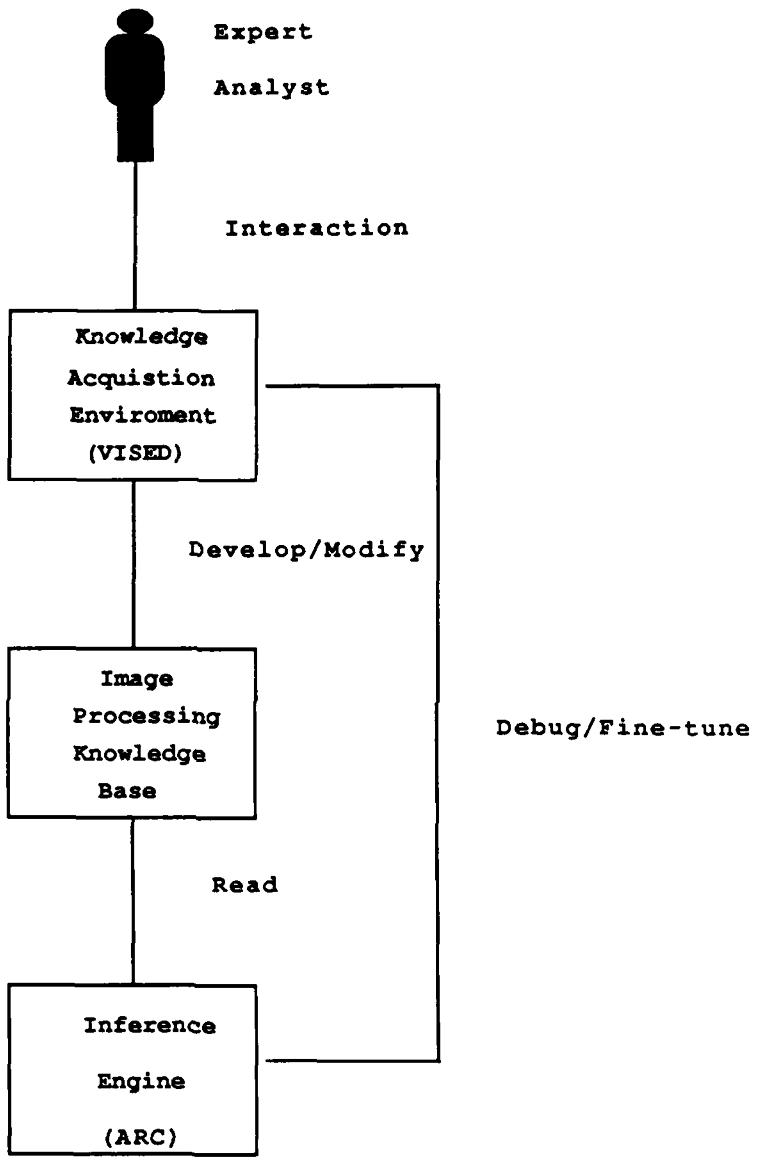
This section describes the system software developed under the project. This software is organized into three major modules: the knowledge acquisition environment, the inference engine, and the knowledge base. Together these modules form an environment for expert system development.

The relationship between the system modules is outlined in figure 3-1. As depicted in the figure, the knowledge acquisition environment is used by the expert analyst to develop and modify the image processing knowledge base. The knowledge base is then read by the inference engine as it develops solutions to image processing problems. If errors are uncovered during the process, the expert reenters the knowledge acquisition environment to debug and fine-tune the knowledge base. Thus, the system defines an environment for interactive knowledge base development.

The relationship between the software developed under this project and a complete expert system are described in the overview of figure 3-2. This figure shows three levels of interaction: interactive knowledge base development, primitive procedure development, and knowledge-based algorithm development. The system constructed under this project creates an environment to support the first type of interaction, namely interactive knowledge base development. However, this is just one aspect of long-term development and use of an expert system. All three interaction levels are required, and figure 3-3 describes how these levels interact over the lifetime of the system. From this figure, it is easy to see that this complete system design supports continued development and maintenance within the familiar rapid-prototype and iterative refinement AI implementation methodology. The software developed under this project has been implemented with this overall scheme in mind and provides the necessary hooks for future development.

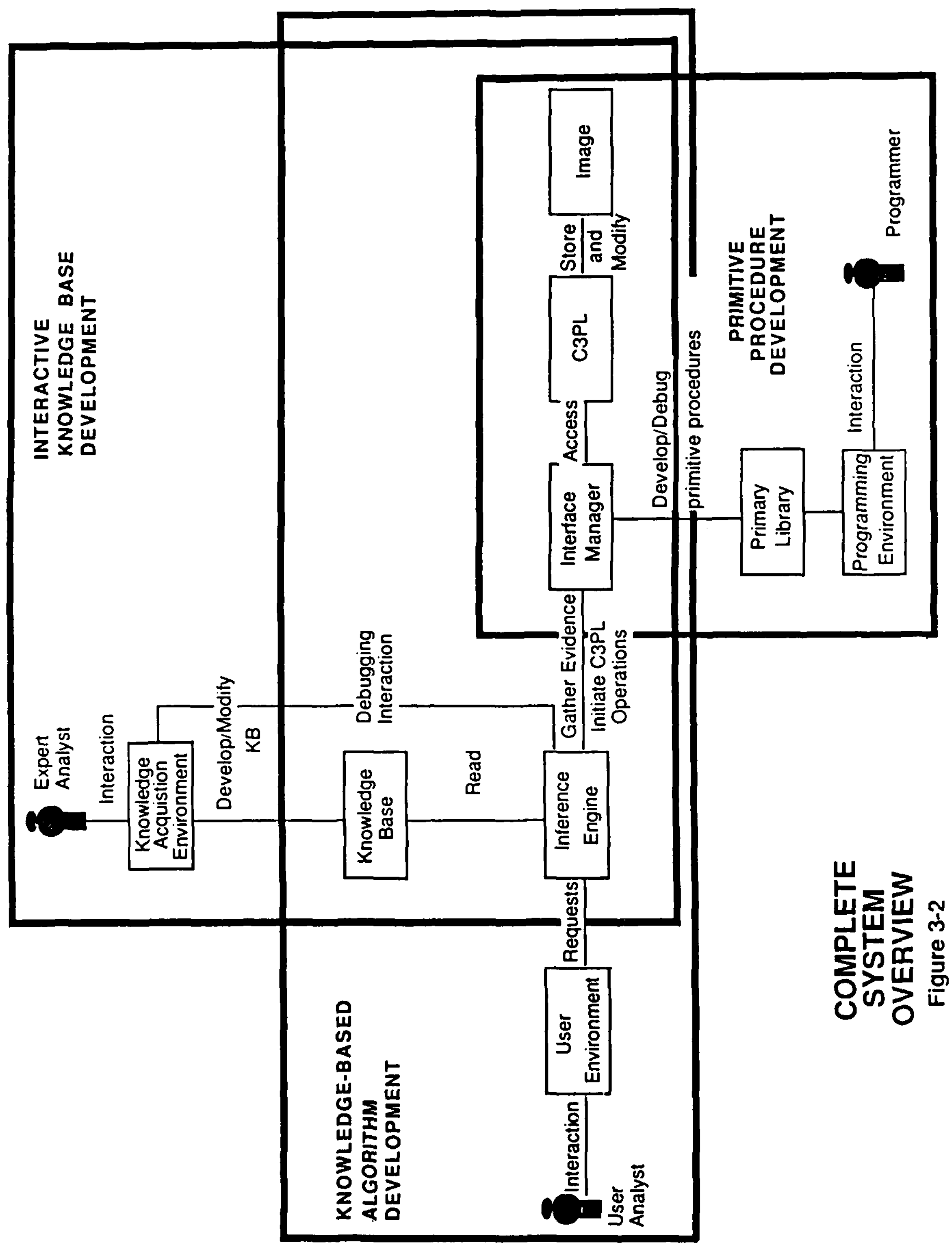
The software developed under this project was implemented on ERIM's Symbolics 3670. It is written entirely in ZetaLisp and Flavors, and takes advantage of many of the features incorporated into the Symbolics environment. In particular, this software relies heavily on the object-oriented style of lisp programming of the flavor system. It also makes considerable use of the Symbolics graphics and user interface components.

The remainder of this section is directed at explaining the system modules in detail.

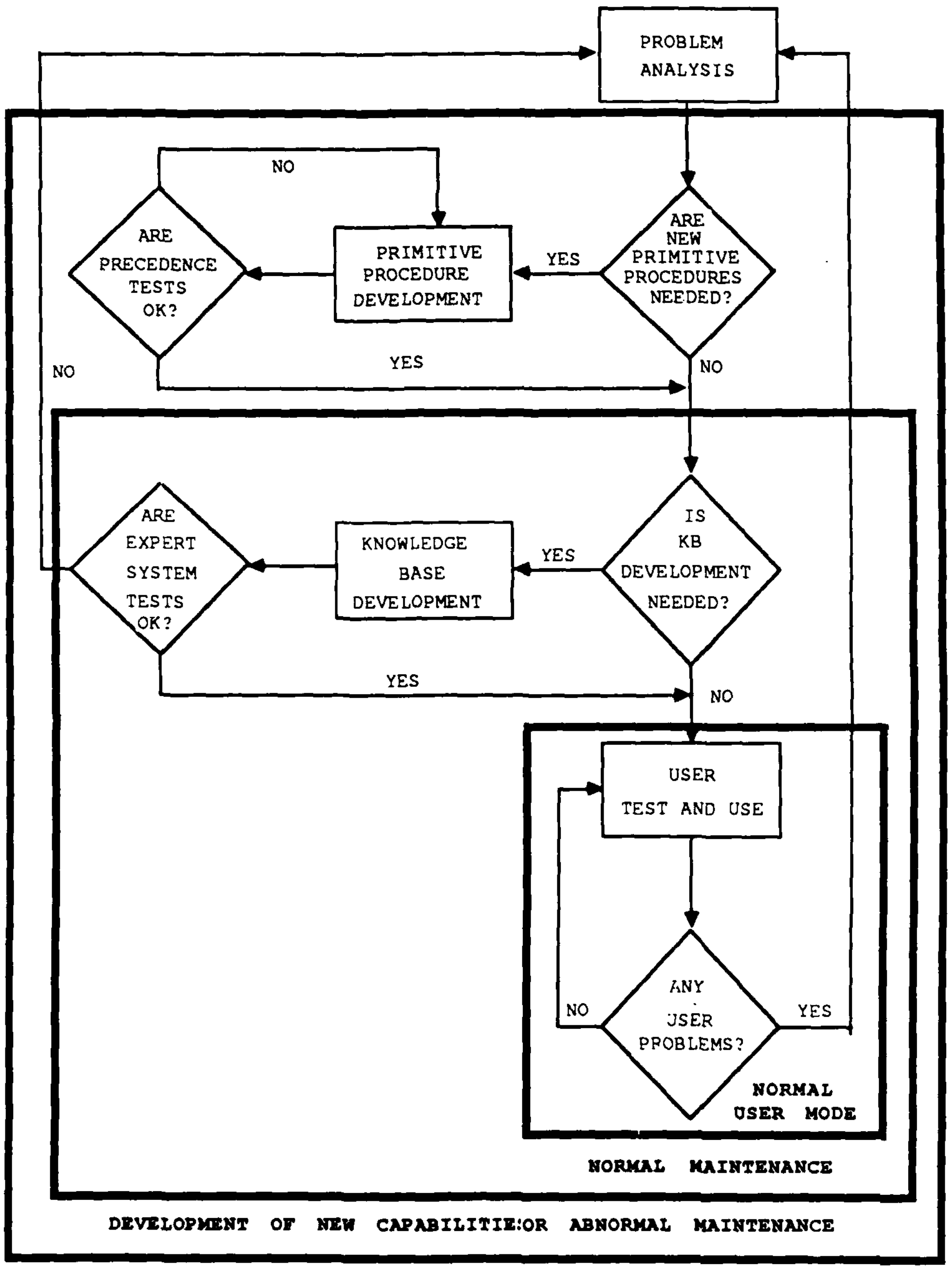


Interactive Knowledge Base Development

Figure 3-1



COMPLETE SYSTEM OVERVIEW
Figure 3-2


PROBLEM ANALYSIS
ARE NEW PRIMITIVE PROCEDURES NEEDED?
YES
PRIMITIVE PROCEDURE DEVELOPMENT
ARE PRECEDENCE TESTS OK?
NO
YES
NO
NO
IS KB DEVELOPMENT NEEDED?
YES
KNOWLEDGE BASE DEVELOPMENT
ARE EXPERT SYSTEM TESTS OK?
YES
NO
USER TEST AND USE
ANY USER PROBLEMS?
NO
YES
NORMAL USER MODE
NORMAL MAINTENANCE
DEVELOPMENT OF NEW CAPABILITIES OR ABNORMAL MAINTENANCE

METHODOLOGY FOR EXPERT SYSTEM DEVELOPMENT/MAINTENANCE

Figure 3-3

3.1 The Knowledge Acquisition Environment

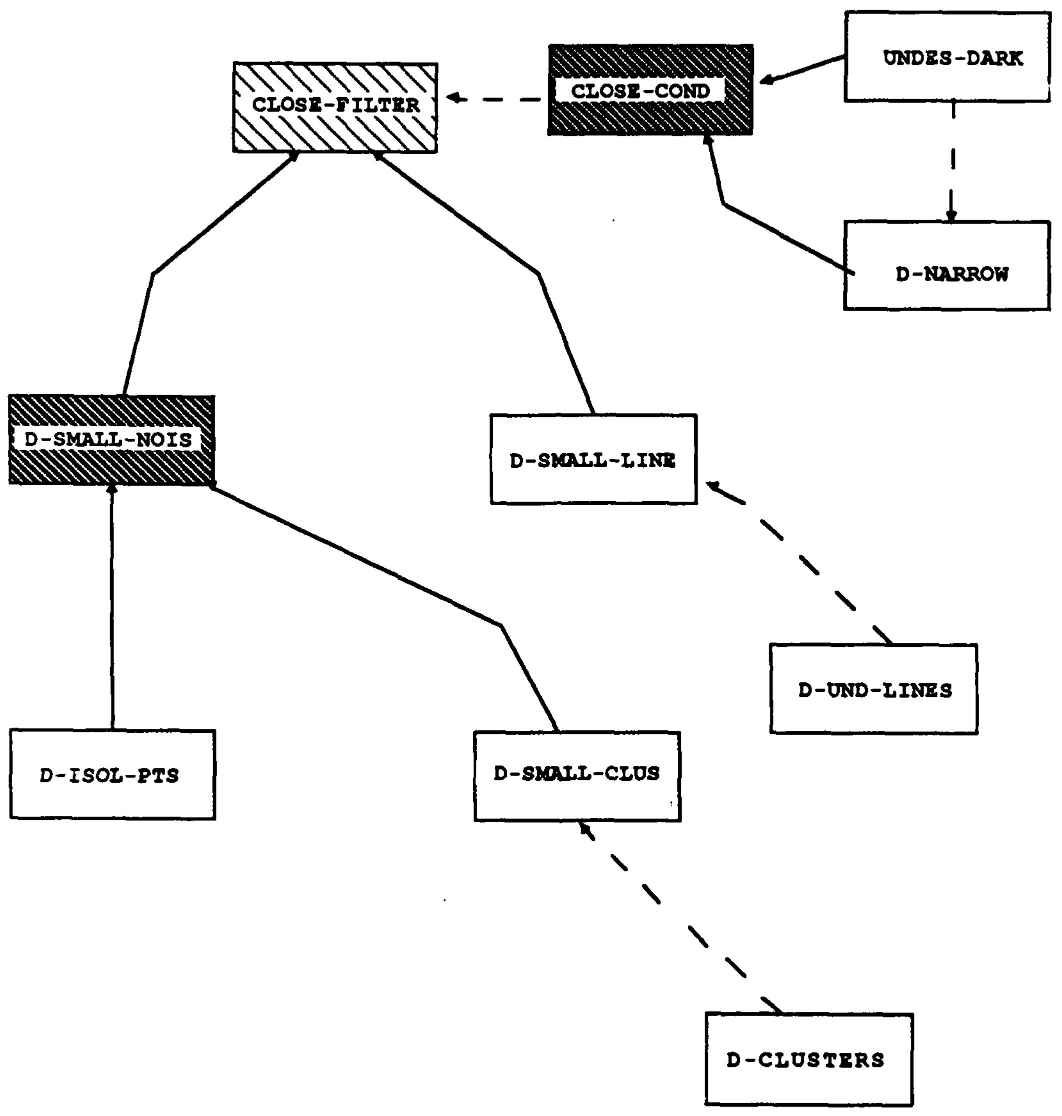
The VisEd knowledge acquisition environment was developed in part under this project. It facilitates rapid development and management of large knowledge base structures. Using the VisEd tool, users can construct and modify knowledge structures with simple mouse and keyboard movements. The resulting knowledge structures can then be evaluated from within the editor, thus enabling rapid, iterative knowledge base development.

The VisEd system was constructed to eliminate much of the wasted effort normally involved in building network structures with natural language oriented representations. These network structures are very common in expert systems, and can be thought of as a collection of nodes. A typical inference network node for knowledge-based image algorithm development would normally be represented in a form similar to the following,

```
(goal          close-filter
 text          "Use closing as a filter."
 anounce       "Considering the use of closing as a filter."
 prior         0.0
 antecedents   (OR d-small-lines d-small-noise)
 context-of    (close-conditions 5 100)
 action        (execute-the-close-filter-operation))
```

In any reasonably sized knowledge-based system the relationships between network nodes is not readily apparent from this syntactic representation. In actual practice, knowledge engineers often draw pictures (e.g. figure 3-4) of the relationships between nodes to fully understand inference network semantics. This practice is both awkward and time consuming. Furthermore, it is not something that can be routinely expected of domain experts when interacting with a knowledge acquisition environment.

The VisEd system enables the user to develop knowledge bases in the more understandable visual form. With this system, the user simply uses the mouse to paint network structures like those of figure 3-5 onto the screen. Once the network structure has been painted, the mouse is used to "open" nodes and links. An example of an "opened" node is shown in figure 3-6. As shown in the figure the "opened" objects display menu forms of object parameters (e.g. text, prior probabilities, etc.) which can then be easily initialized or updated. Error checking has been built into the system and is continuously performed to assure that the semantic integrity of the network is maintained. Prompts are given on the screen to assist the user in selecting the proper values for all parameters.



INFERENCE NETWORK EXAMPLE

Figure 3-4

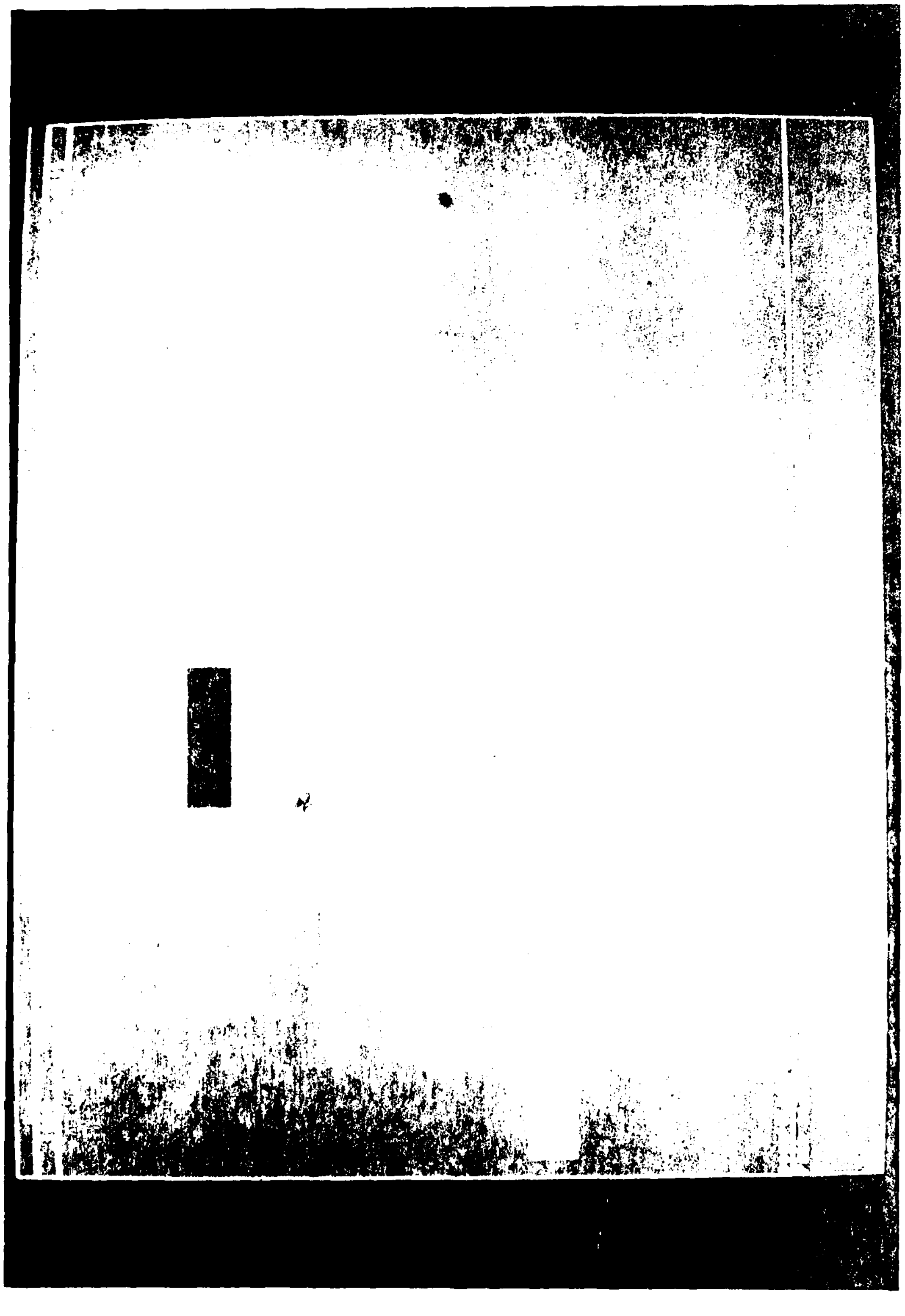
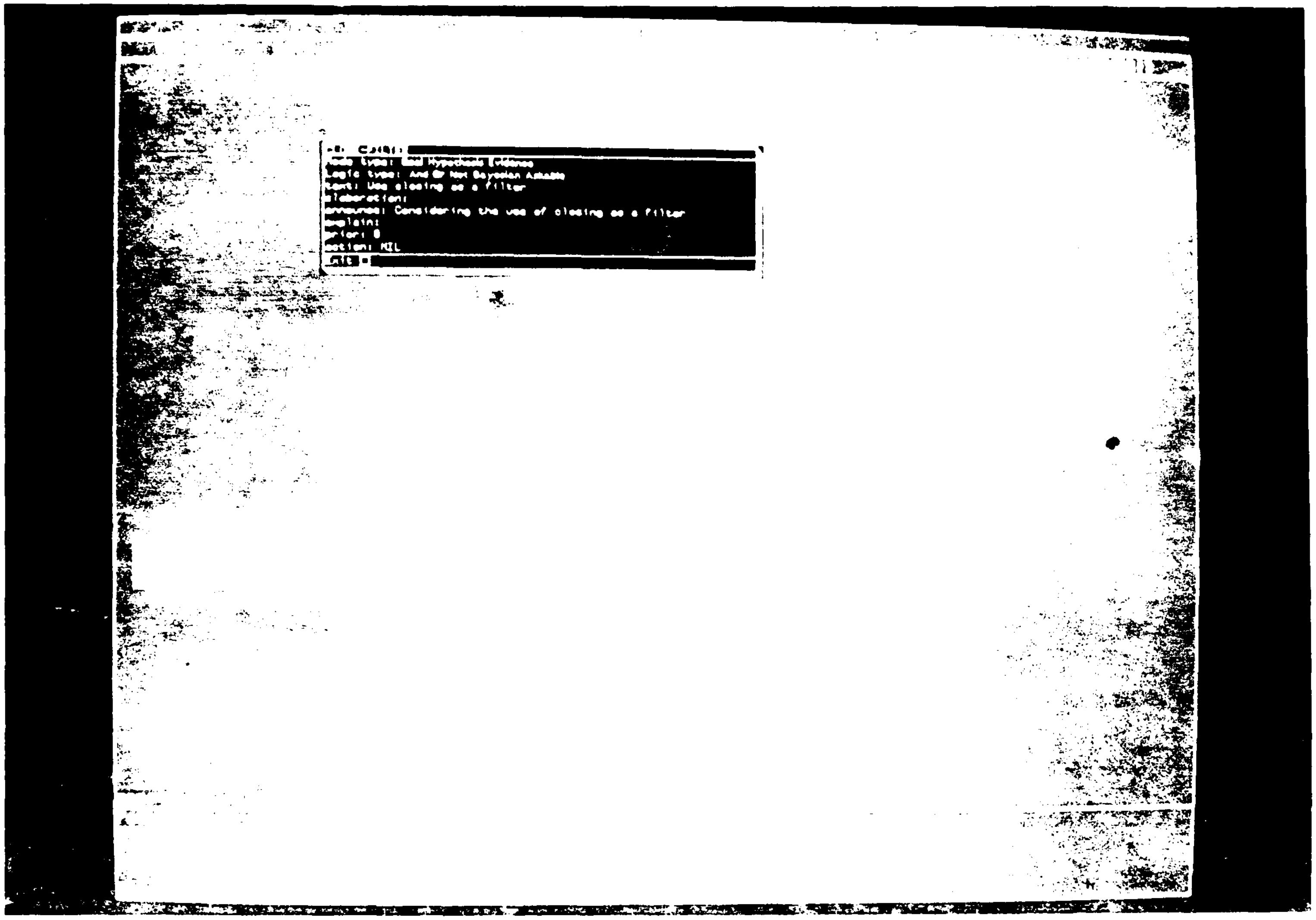


Figure 3-4



```
[illegible]
Node type: [illegible] Hypothesis Evidence
Logic type: And Or Not Bayesian Askable
Text: Use closing as a filter
Elaboration:
Announce: Considering the use of closing as a filter
Explain:
Prior: 0
Action: NIL
Exit
```

Figure 3.6.

VisEd uses multiple pages with indexing and windowing to further reduce the overall complexity of the system. This enables the user to develop a knowledge base in book like form. Network structures are painted on pages. Individual nodes may occur on more than one page. Any *modifications* to these *multiply* represented nodes are reflected in all occurances. Movement around the network is mouse controlled and includes simple page turning, individual page scrolling, and page lookup through a mouse sensitive page name index.

VisEd can automatically produce a knowledge base in a variety of different syntactic forms for input into inference engines. Since knowledge about these syntactic representations is incorporated into VisEd, this removes the burden of dealing with these forms from the user. Thus, the user is not required to deal with lisp structures, parenthesis counting, syntactic editors, etc. Currently, VisEd supports three different forms: an internal binary form, a Lisp form, and a Vax form. Additional forms can be easily incorporated into the system to facilitate use with other expert system shells.

VisEd also features very flexible and extensible capabilities. Domain specific primitives can be added to a library of primitives to enable use of this approach to acquiring knowledge across a very broad field of complex domains. This enables the development and maintenance of intelligent systems in areas that require a mixture of both conventional and Artificial Intelligence approaches. This layered approach partitions the development modes of the system in such a way that domain experts with no previous computer experience can build and maintain knowledge structures.

The VisEd system is not limited to use on this project and can be used to create expert systems in a wide range of problem domains. It has proved to be an extremely valuable tool for knowledge development and maintenance. It has proved to be an especially valuable tool for refining and maintaining expert system knowledge bases.

3.2 The Inference Engine

The Automated Reasoning Component (ARC) has been developed by ERIM in part for use in this project. This inference engine is much like the one found in the classic expert system, PROSPECTOR. It is a goal-directed system with both fuzzy logic and Bayesian uncertainty mechanisms.

With ARC, as in PROSPECTOR, rules are used to form an inference network structure. This structure contains top-level hypotheses, called goal hypotheses, which are decomposed into various levels of subhypotheses. The subhypotheses are further broken down into lower level subhypotheses until eventually they become specific items of evidence. With each node, there is an associated prior degree of belief and a rule for combining subnode degree of belief into an updated degree of belief for the node.

During a consultation session, ARC attempts to evaluate the degree of belief of a goal hypothesis by chaining down the inference net, identifying the evidence items that affect the goal hypothesis, and querying the user or a database for each relevant item of evidence. Evidential degrees of belief are then propagated through the inference network to determine whether or not a goal is accepted.

3.2.1 Inference Network

Knowledge is represented in the ARC system in the form of an inference network. This is a directed graph comprised of nodes and links. The nodes in the network represent individual antecedent conditions, consequent conditions, or context conditions. The links represent either logical or contextual relationships between nodes.

Three different node types may be contained in the network structure: goal, hypothesis, and evidence. The goal nodes are generally the root vertices of the network, but not necessarily. They represent the propositions that the system is attempting to conclude. The hypothesis nodes are intermediate vertices in the network structure. Generally they are intermediate propositions that the system must resolve in order to infer something meaningful about the system goals. These nodes are typically identified as those nodes that are neither goal nodes or evidence nodes. Evidence nodes are simple propositions whose status can be determined by either querying the user, an external database, or a sensor. Generally these nodes are the leaves of the network (i.e. nodes with no links from any nodes representing antecedents.) Associated with each node are individual node attributes. These attributes bind various additional information to the nodes.

Two general classes of link types may be contained in the network structure: logical and context. The logical links connect antecedent conditions with their consequents. Within the class of logical links there are four different instances: AND, OR, NOT, and BAYESIAN. These different instances determine the way degrees of belief are combined and assigned to higher level nodes within the network structure. The context links indicate that one proposition establishes the context within which the other is explored. Certain logical and context links are also associated with individual link attributes. The context links have associated lower bound (LB) and upper bound (UB) attributes that establish a range under which context belief requirements are satisfied. The Bayesian links have positive (PW) and negative (NW) weights that establish the maximum effect that the antecedent evidence can have on the consequent node.

Figure 3-4 provides an example of an ARC inference network structure. In this example the the close-filter node is a goal, the close-cond and d-small-nois nodes are subhypotheses, and the remaining nodes are evidence nodes. In this figure the dark lines represent logical relationships between nodes and the dashed lines represent contextual relationships.

3.2.2 Uncertainty Mechanism

Each node in the inference network has a prior and current degree of belief associated with it. The degree of belief ranges from -100.0 (false) to +100.0 (true), where 0 represents uncertainty. Initially, the current degree of belief for a particular node is the same as its prior degree of belief.

Degrees of belief are equivalent to probabilities. The relationship between the two is defined as

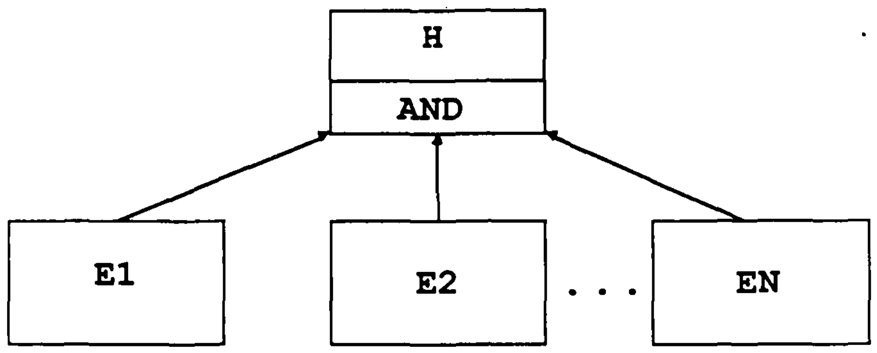
$$Deg = 10 * \log10 (p/(1-p)).$$

Thus, degrees of belief are just ten times the log base ten of the odds function. This degree of belief scale is arbitrary, but psychological arguments can be made for its use.

The uncertainty mechanism for the system is a combination of fuzzy logic and Bayesian reasoning. The combination formulas of the AND, OR, and NOT rules are based on ideas from fuzzy logic. The combination formula for the Bayesian rule is based on Bayes rule with independence assumptions.

The rules for combining evidence within this structure are defined in figures 3-7 through 3-10. Thus, for an AND rule the degree of belief associated with the consequent is the minimum of the degrees of belief of the antecedents. For the OR rule the degree of belief is the maximum of the degrees of belief of the antecedents For the NOT rule, the degree of belief is the negative of the belief of the antecedent. For the Bayesian rule, the degree of belief of the antecedent is the current belief plus

IF E1 AND E2 AND . . . EN THEN H

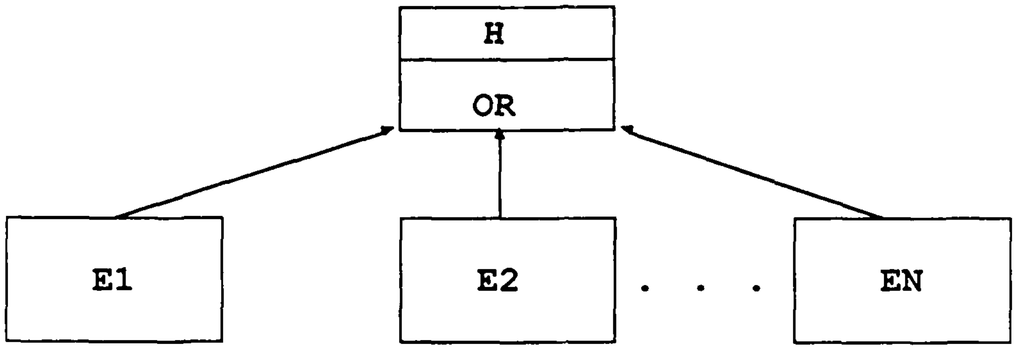


BELIEF (H) = Min { BELIEF (Ei) }
1 <= i <= N

Degree of Belief Propagation For AND Rules

Figure 3-7

IF E1 OR E2 OR . . . EN THEN H

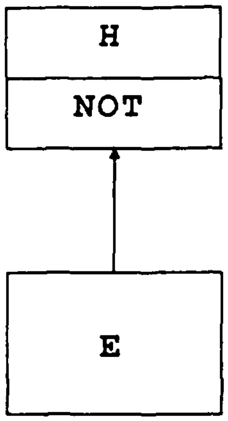


BELIEF (H) = Max { BELIEF (Ei) }
1 <= i <= N

Degree of Belief Propagation For OR Rules

Figure 3-8

IF NOT E THEN H

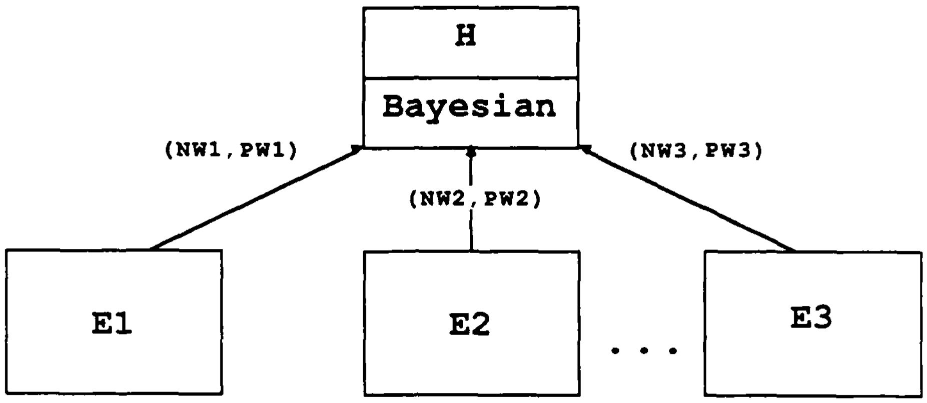


BELIEF (H) = - BELIEF (E)

Degrees of Belief Propagation for NOT Rules

Figure 3-9

IF BAYESIAN (E1, E2, . . . , EN) THEN H



$$\text{BELIEF}(H) = \text{PRIOR}(H) \sum_{1 \le i \le N} \text{WEIGHT}(E_i), \text{ where}$$

$$\text{WEIGHT}(E_i) = \begin{cases} \text{Sign}(PW) * \min' PW \,|\, , & \text{Belief}(E_i) - \text{PriorBelief}(E_i) > \text{Prior}(E_i) \\ 0 & ; \text{Belief}(E_i) > \text{Prior}(E_i) \\ \text{Sign}(NW) * \min' NW \,|\, , & \text{Belief}(E_i) - \text{Prior}(\text{Belief}(E_i) > \text{Prior}(E_i) \end{cases}$$

Degree Of Belief Propagation for Bayesian Rules

Figure 3-10

the sum of the evidence weightings determined by linear interpolation between the positive and negative weights.

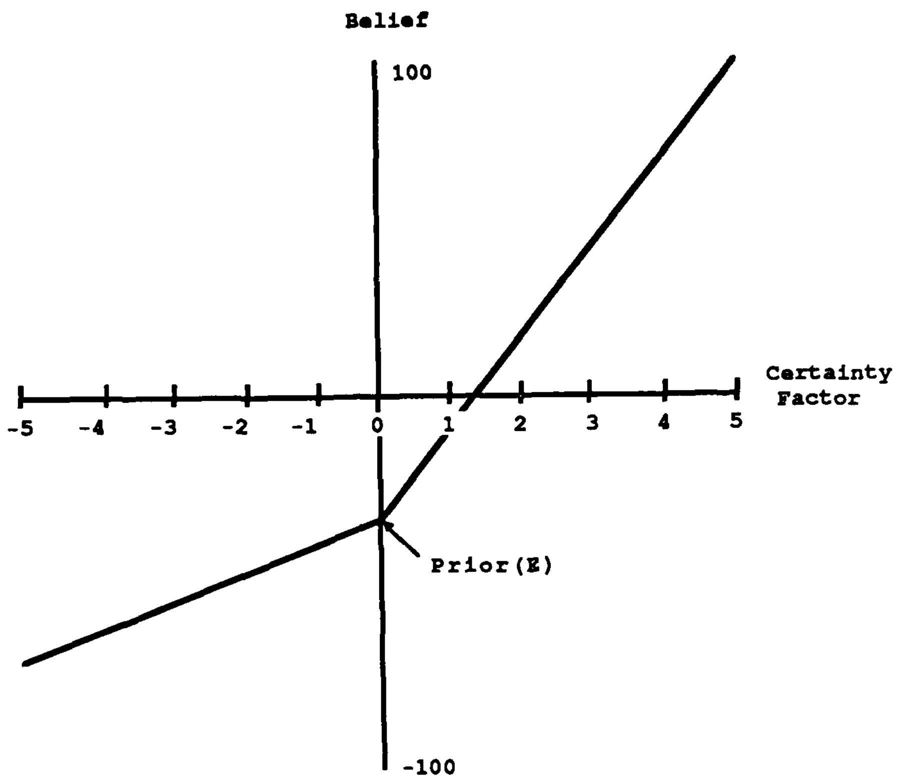
Using these combination rules, belief is propagated through the network structures. Evidence is incorporated into the system through the evidence nodes. This evidence is represented in the form of degrees of belief. It is usually obtained through dialog with the user or through querying some existing data base. The evidential degrees of belief are then propagated through the network structure to update the degrees of belief associated with consequent hypotheses. The updating rules outlined above are used to determine the new degrees of belief for these consequent hypotheses.

Evidence is introduced into the system in the form of certainty factors. These factors are real numbers that range from -5.0 (absolutely false) to +5.0 (absolutely true) with 0.0 corresponding to uncertainty. Thus, values from -5.0 to 0.0 indicate some certainty that the assertion is false and values from 0.0 to +5.0 indicate some certainty that the assertion is true. These certainty factors are convertered into degrees of belief through piecewise linear interpolation. The interpolation formulas are illustrated in figure 3-11. From this figure it is easy to see that if the user inputs a certainty factor between -5.0 and 0.0, then the computed degree of belief will be between -100.0 and the prior degree of belief. Similarly, if the user inputs a certainty factor between 0.0 and +5.0, then the computed degree of belief will be between the prior degree of belief and +100.0. It is worthwhile to note that under this scheme a certainty factor of zero is converted into the prior degree of belief. Thus, if a user enters a zero (no knowledge), the belief defaults to the prior degree of belief.

3.2.3 Inference Mechanism

The ARC system reasons by propagating degrees of belief through the inference network structures created by VisEd. ARC begins this process by selecting the largest, currently uninvestigated, goal from the set of inference network goal nodes. Once it has a goal to consider, ARC will select and examine evidence for that goal until there is no more evidence to consider.

Evidence is selected by recursively examining the inference net links to the goal node until evidence nodes are found which, when evaluated, may have some effect on the degree of belief of the current goal. The evidence selection mechanism considers all yet to be evaluated evidence for the goal, and selects the piece of evidence that could cause the largest local shift in the degree of



$$\text{Belief}(E) = \begin{cases} \dfrac{(\text{Prior}(E) + 100)(\text{Certainty}(E) + 5)}{5} - 100 & ; \quad \text{Certainty}(E) < 0 \\ \text{Prior}(E) & ; \quad \text{Certainty}(E) = 0 \\ \dfrac{(\text{Prior}(E) - 100)(\text{Certainty}(E) - 5)}{-5} + 100 & ; \quad \text{Certainty}(E) > 0 \end{cases}$$

Certainty Factor to Belief Conversion
Figure 3-11

belief associated with the goal. Thus, for AND nodes, the smallest uninvestigated antecedent of the current node is selected for consideration. For OR nodes, the largest uninvestigated antecedent of the current node is selected. For NOT nodes the investigated antecedent is selected. For the Bayesian nodes the uninvestigated antecedent with the largest absolute PW or NW weight is selected.

Context links effect the evidence selection process. Context links take precedence over logical links, and thus, evidence nodes that are associated with the goal node through context links are explored first. Once the portion of the inference network below an antecedent node of a context link has been fully explored it is labeled investigated and given a final degree of belief. This belief is then compared with the [LB,UB] interval of the context link to determine if the context has been satisfied. If the final belief fails to fall within the interval, then the consequent context node is labeled fully investigated and warrants no further exploration. If the final belief falls within the interval the context is established, and the portion of the network below the consequent context node is opened for exploration. Evidence below this node is then selected in the usual manner.

Once an evidence node is selected for exploration, it is triggered. If it has a primitive function located in its action slot, then that function is attached and evaluated. If not, then the sentence in the text slot is used to create a user prompt. In either case, the response is a certainty factor which is immediately converted into a degree of belief. This belief is then recursively propagated upward through the network, thus updating the belief associated with all hypotheses effected by this new evidence.

The process of selecting, triggering, and propagating is continued until the goal becomes investigated. This can happen in several ways. If all the antecedent nodes of a goal are explored, then a final degree of belief can be associated with the goal, and hence, it is fully investigated. If the network reaches a state in which an exploration threshold is exceeded and indicates that no significant impact can be made even if the remaining uninvestigated evidence nodes are explored, then further analysis of the goal becomes uninteresting, and thus, the goal is labeled investigated. If critical context requirements are not met, then the goal is viewed as not worth exploring and is labeled investigated. Various combinations can also occur, but all result in a final degree of belief for the goal node.

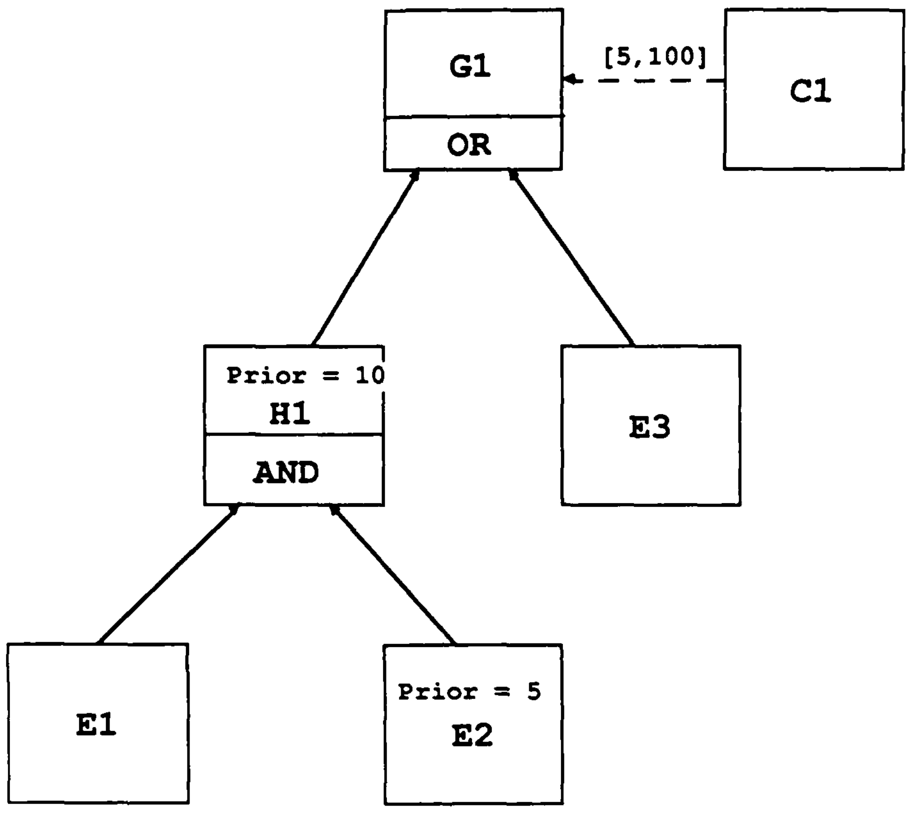
Once the goal has been labeled investigated, it is analyzed to determine whether it has been accepted or not. This is determined by simply comparing its final degree of belief with a belief threshold. If the goal is accepted, the user is queried to determine whether the action associated with the goal should be triggered. If the answer is no, then ARC continues processing goals. If the answer is yes, then the procedure located in the goal action slot is attached and evaluated. Since this action is effectively a conventional procedure call, this facility can potentially trigger a wide range of outcomes. After the attached procedure is completed, ARC prompts the user to determine if continued reasoning is required. If so, ARC continues processing goals.

ARC continues selecting and processing the largest uninvestigated goal until one of two situations occur. Either a satisfactory goal is obtained and the user indicates that further exploration is unnecessary, or ARC runs out of goals to process.

To clarify the ARC inferencing mechanism, consider the small hypothetical example illustrated in figure 3-12. As illustrated in the figure the example network is composed of six nodes. Node G1 is the goal. Node H1 is a hypothesis node. Nodes E1, E2, and E3 are evidence nodes. Node C1 is an evidence node that establishes context for G1. If either H1 OR E3 are true then the goal, G1, is true. If both E1 AND E2 are true, then H1 is true.

The following dialog traces the reasoning that ARC performs to combine information according to the example network structure.

```
(01) New goal selected is G1

(02) Chaining on G1

(03) Chaining on C1

(04) New evidence selected is C1

(05) Enter belief for C1 : 4

(06) Updating current belief of C1 from 0 to 9.542424

(07) Marking C1 investigated.

(08) Chaining on G1

(09) Chaining on H1

(10) Chaining on E1

(11) New evidence selected is E1
```



Uncertainty Propagation Example

Figure 3-12

```
(12) Enter evidence for E1 : 4

(13) Updating current belief of E1 from 0 to 9.542424

(14) Updating current belief of H1 from 10 to 5

(15) Updating current belief of G1 from 0 to 5

(16) Marking E1 investigated.

(17) Chaining on G1

(18) Chaining on H1

(19) Chaining on E2

(20) New evidence selected is E2

(21) Enter evidence for E2 : -3

(22) Updating current belief of E2 from 5 to -3.5994349

(23) Updating current belief of H1 from 5 to -3.5994349

(24) Updating current belief of G1 from 5 to 0

(25) Marking E2 investigated.

(26) Marking H1 investigated.

(27) Chaining on G1

(28) Chaining on E3

(29) New evidence selected is E3

(30) Enter evidence for E3 : 3

(31) Updating current belief of E3 from 0 to 6.0206003

(32) Updating current belief of G1 from 0 to 6.0206003

(33) Marking E3 investigated.

(34) The G1 does not look promising

(35) Marking G1 investigated.

(36) No goals left to process.
```

(37) Reasoning completed!

The reasoning on this example structure begins with ARC selecting G1 for consideration (01). ARC the searches for evidence to consider for G1 (02)-(03). C1 is selected first, becuase it is required to establish the required context for G1 exploration (04). Evidence for C1 is obtained (05). It is converted into degree of belief and updates the current belief associated with C1 (06). C1 is marked as being fully explored (07). Since the new belief for C1 falls within the [LB,UB] bounds, the context for G1 exploration is satisfied. ARC continues to search for evidence to support G1 (08)-(10). E1 is selected as the next best evidence to explore (11). Evidence is obtained (12) and used to update the degree of belief for E1 (13). This evidence is then propagated through the network (14)-(15). The degree of belief for H1 is changed (14) from its prior belief, 10, to the minimum belief currently associated with E1, 9.54, and the prior belief of E2, 5. The belief of G1 is changed (15) from its prior belief to the maximum of the current belief of H1 and the prior belief of E3, 0. E1 is now considered fully investigated (16). ARC continues to search for more evidence to consider (17)-(19). It selects E2 (20). It obtains new evidence (21) and converts it to the degree of belief for E2 (22). This new evidence is propagated through the network (23)-(24), and E2 is marked as being fully explored (25). Since there is no remaining evidence below H1, it too is marked as being fully explored (26). ARC continues to search for evidence (27)-(28). It selects E3 (29), obtains new evidence (30), and converts this new evidence to degrees of belief (31). It propagates this evidence (32). E3 is now fully explored (33). Now G1 is fully explored, but does not have enough support to be considered as satisfied. The user is informed (34), and the goal is marked as being fully explored (35). ARC continues to search for goals to consider. There are none left to explore (36). ARC is finished with this example (37).

3.3 Knowledge Bases

A filter knowledge base was developed under the fourth phase of the project. This knowledge base is directed at providing the information required to satisfy the first phase, background/noise removal, of the edge extraction algorithm sequence as outlined below in the system design section. Thus, this work can be viewed as the first step in constructing the knowledge structures required for the image algorithmist aid for edge extraction as defined under this project.

The filter knowledge base currently contains information about seven C3PL filters: the OPEN-FILTER, CLOSE-FILTER, DONUT-FILTER, HOLE-FILTER, SPIKE-FILTER, BUMP-FILTER, and POTHOLE-FILTER. The knowledge contained within this data structure is far from complete. It is, however, fairly representative of the knowledge required to support this phase of the overall edge extraction sequence. It is also remarkably robust, especially considering the small amount of effort that was allocated to this subtask under the project.

Additional knowledge base development is expected under subsequent program phases. This development is expected to proceed in a planned, programmatic fashion as outlined in the system design section. It is also expected that some experimentation will be required to fully determine the image conditions under which infrequently used operations best perform.

4 DISCUSSION OF RESULTS

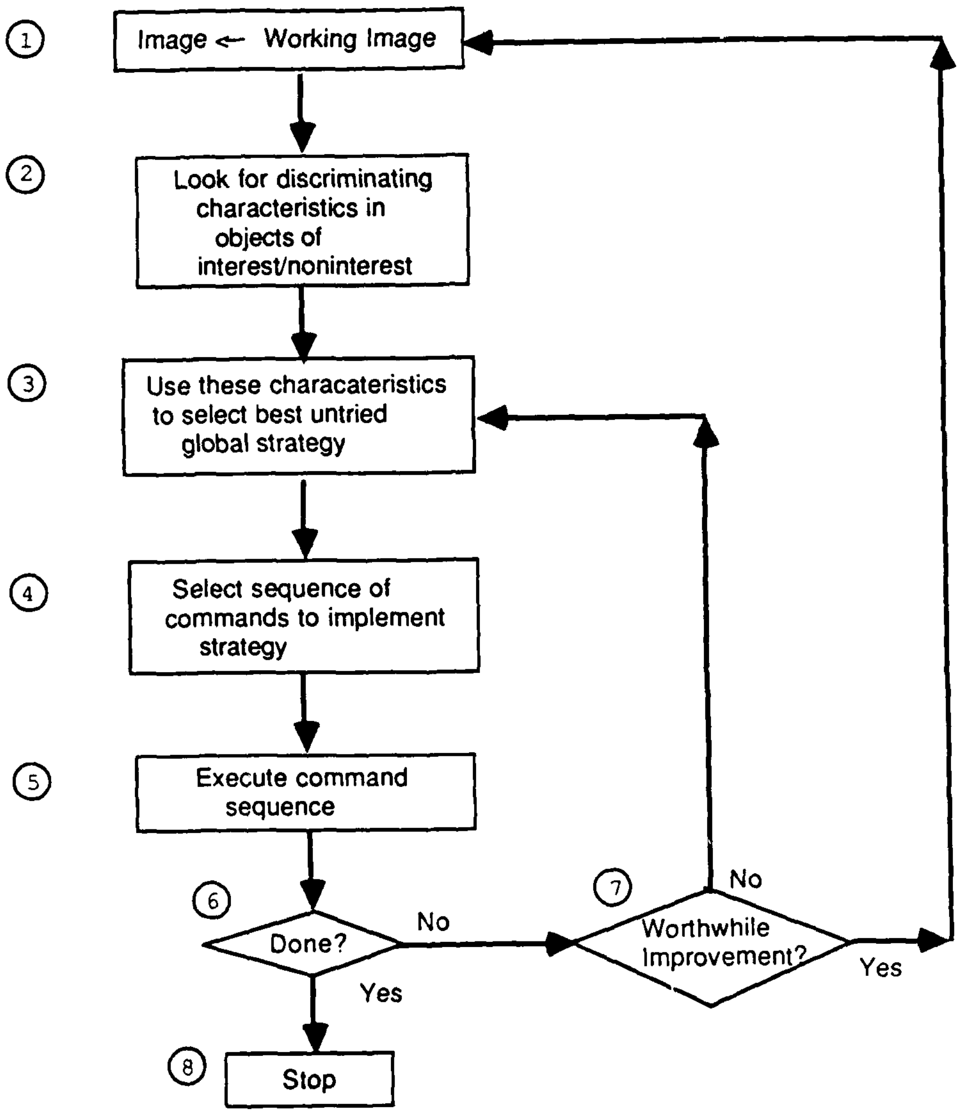
Several positive outcomes have resulted from the research conducted under this effort. These outcomes include a much deeper understanding of the image algorithm development process, an evolving image processing vocabulary, a collection of global algorithm strategies, an overall system design for an image algorithmist aid for edge extraction, an implemented knowledge acquisition environment, an implemented inference engine, and a working prototype expert system.

The remainder of this section is directed at describing these outcomes in detail.

4.1 Understanding Of Image Algorithm Development

This project has resulted in a greater understanding of the process of image algorithm development. At the onset, it was generally conceded that algorithm development was a complex task. It was realized that such development is a highly experimental process as characterized in figure 4-1. This view has not changed. What has changed, however, is that now much more is known about the complexities associated with this task. In light of this new understanding, the remainder of this section is spent in discussing some of the major issues associated with developing image algorithmist expert systems.

Large amounts of quantative information are required to develop image algorithms. Much of the time spent in the initial knowledge engineering sessions was focused on counting pixels.


1
Image <- Working Image
2
Look for discriminating characteristics in objects of interest/noninterest
3
Use these characateristics to select best untried global strategy
4
Select sequence of commands to implement strategy
5
Execute command sequence
6
Done?
No
Yes
7
Worthwhile Improvement?
No
Yes
8
Stop

Expert Image Algorithm Development

Figure 4-1

Edge widths, noise diameters, spacing sizes, etc. are all important facts that are used not only in selecting command parameters but also in selecting the commands themselves.

Numerous global strategies for algorithm development exist. Some of these strategies are straight forward. Some of them are tricky and clever. Not all algorithmists have the same set of strategies to choose from. They are usually exchanged through word of mouth or through technical papers. No apparent work has been done to collect these strategies for instructional purposes.

Multiple solutions to image problems are common. For example, consider the following prototypical problem

ABSTRACT EXAMPLE PROBLEM

ACCEPT Set	REJECT Set
Pulse Lines	Pulse Lines
Broken	Broken
Straight	Curved
X-Y Oriented	

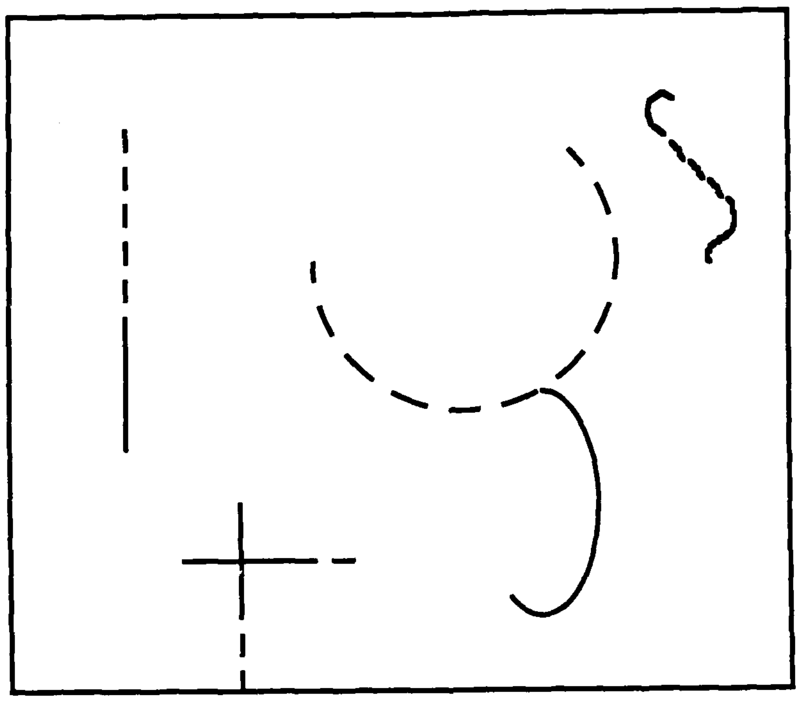
Figure 4-2 represents a typical image for this problem. The task is to filter out the curved lines. Several potential solutions to the problem exist. As an example of this, consider the following four solutions.

Strategy 1: Join up ALL lines, with preference for X-Y directions, then find the straight ones of sufficient size.

Algorithm:
Union of closings in X/Y directions, followed by union of openings in X/Y directions, both with straight seg-ent structuring elements.

Parameters:
Choose length of closing segments as Maximum of largest X/Y Gaps which need to be covered. Length of opening segment should be larger than maximum of closed "reject" segments, and smaller than minimum of closed "accept" segments.

Caveats:
-Closings may join up other parts of image, particularly middle regions of lines which are too close together.
-Ranges of gap lengths and edge lengths may make it not possible to do a complete separation this way.



Problem : Find the Straight X-Y Oriented Lines

Prototypical Image Processing Problem

Figure 4-2

Strategy 2: Find only the Ends of lines (especially only the ends of horizontal or vertical segments), join these up, add to original image, then select straight X/Y segments.

Algorithm:
Union of Openings by X/Y segments to select H/V lines (=I1)
Find end points of I1 segments (=I2) (Match operation)
Union of Closings of I2 by straight H/V segments. (=I3)
I4 = I1 Union I3
Union of Openings of I4 by straight H/V segments (Done).

Parameters:
Same considerations as above.

Caveats:
-Ranges of accept vs reject gaps and segments after closing may not be completely disjoint.
-Eight-connectivity problem--how do we handle end points which are diagonal, or do we ignore?

Strategy 3: After connecting up gaps, find components which are not H/V, and use span/cdilate to recover their length and then reject them.

Algorithm:
Join up as before. (=I1)
Union of erosions by Non-H/V elements to find zones which are not H/V.
Conditionally dilate these over I1 to get those which contain non-H/V elements.
Subtract result from I1 to get just H/V lines.

Parameters:
Same as above for joining up operation.

Caveats:
Will not work if curved and straight elements cross, or touch.

Strategy 4: If curved lines cross straight ones, then find 3 and 4 connection points, and block the conditional dilation of non-H/V segments at these points, to keep them from spilling into straight H/V segments. Put them back in after the lines containing non-H/V zones have been removed.

Solving even small prototypical problems can be a very involved process. This is very apparent in the examples above. It also gives some insight into how difficult it really is to develop algorithms for complex images.

Experimentation is characteristic of algorithm development. Most often algorithmists, even expert algorithmists, must resort to trying various combinations of operations before selecting the appropriate approach. Since this is an inherent aspect of algorithm development, any expert system developed in this area must incorporate this aspect into its system design.

The experimentation aspect of image algorithm development is often coupled with distinct algorithm phases. Within each of these phases there is definite experimentation, but between each of the phases there is little change or modification. This aspect of algorithm development is examplified by the typical filtering phase that most often initiates algorithm development. Under this phase as much unwanted noise is removed as possible. It produces a filtered image that serves as sort of an anchor point in the overall algorithm development. It is rarely modified further, and serves as the fixed input to further algorithm development. This aspect of algorithm development is important in that it identifies valuable information for expert system design.

Knowledge of sensor characteristics is important in developing image algorithms. The sensor that was used to acquire an image has a definite effect on the characteristics of an image, and an understanding of this effect must be incorporated into image algorithms. This is especially true when the algorithm is expected to work over class of images. For example if a sensor is known to produce certain noise patterns, then this information should be incorporated into the filtering phase of algorithm development.

Many aspects of the image algorithm development process are very much knowledge intensive. Referring back to figure 4-1, almost every step of the algorithm involves a great deal of knowledge. In fact, in many of the steps, there exist several possibilities for developing expert systems to assist in the process. Understanding the depth of the knowledge involved in the overall task is extremely valuable and critical to developing the scope for expert system development in this area.

The increased understanding of the image algorithm development process that has resulted from this project is important for several reasons. First, it provides a great deal of valuable information about how to train and develop new image algorithmists. Second, it provides the necessary insight into the problem required to build intelligent tools to support the algorithm development. Third, and last, it also provides insight into what conventional software tools also need to be developed to support the process.

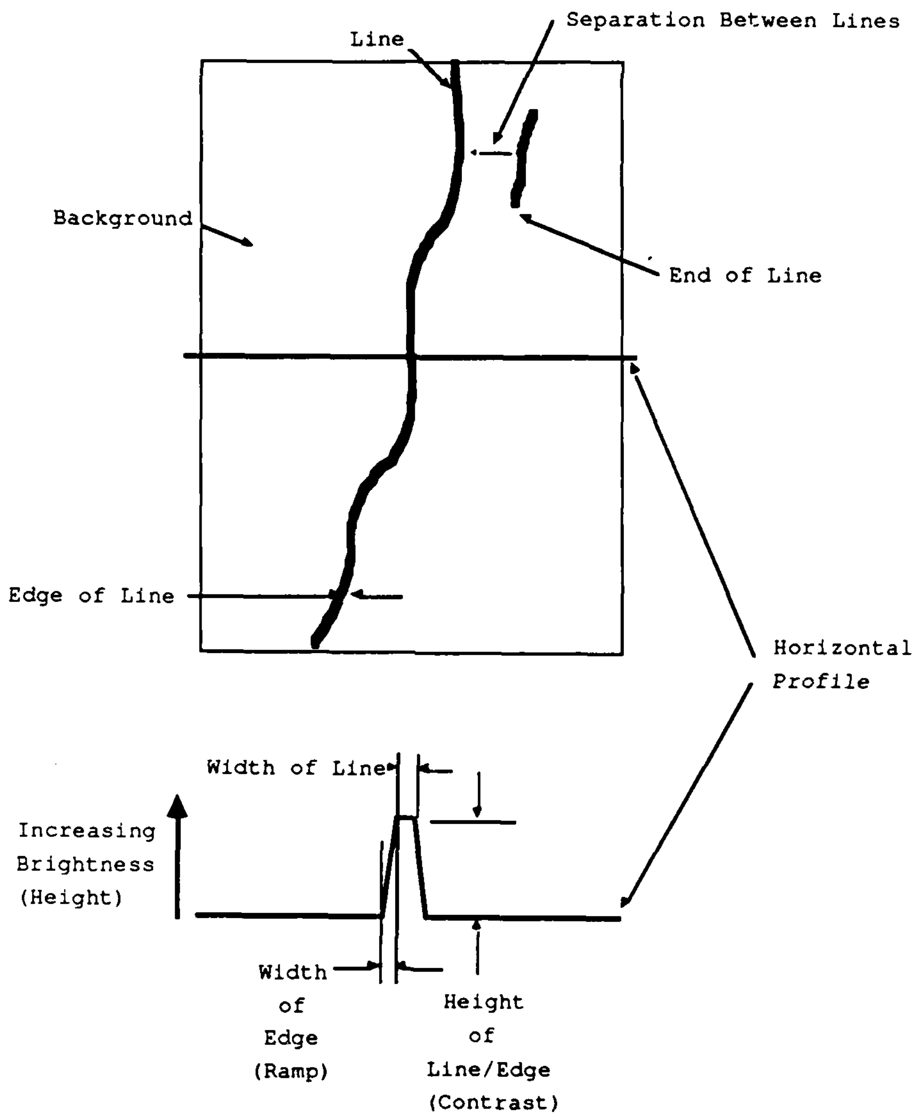
4.2 Vocabulary Of Image Processing Terms

This project has resulted in the beginnings of a vocabulary for discussing image processing ideas and concepts. From the very first knowledge engineering session it was apparent that a vocabulary for discussing image processing concepts was needed. In fact, the early sessions were characterized by a great deal of effort spent in defining and explaining terms and ideas. This work was routinely documented and catalogued to produce working vocabulary. This vocabulary was then used to produce more efficient and productive subsequent sessions. Examples of the vocabulary are shown in figures 4-3 and 4-4.

Again, these results have importance that exceeds this particular project. Since this vocubulary has produced much more direct communication of image processing ideas in knowledge engineering sessions, it can also be used to produce more efficient knowledge transfers between algorithmists at all levels.

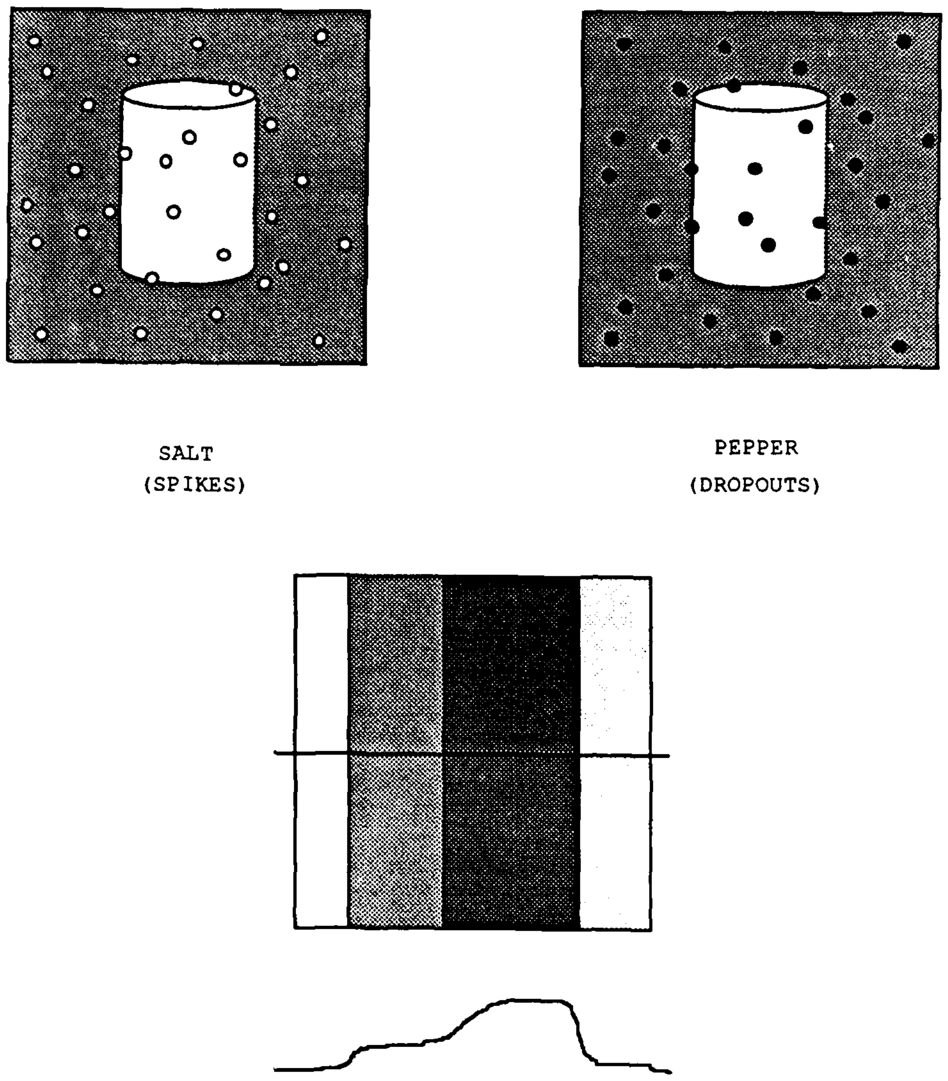
4.3 Global Strategies

This project has resulted in the cataloging of several global strategies used by image algorithmists when developing algorithms. The findings from this work is organized into three subsections below. In the first subsection, strategic issues for separating desired from undesired edges are explored. In the second subsection general approaches for developing algorithms on different classes of lines are presented. In the last and third subsection, suggested methods for specific edge types are presented. The compiled information presented in these subsections is in itself very valuable, especially when viewed as guidelines for developing and training new algorithmists.



Vocabulary Example - GENERAL TERMS

Figure 4-3



Vocabulary Example - NOISE

Figure 4-4

4.3.1 Strategic Issues In Separating Desired From Undesired Edges

This general strategy has to do with eliminating undesired information from an image by using some criterion or combination of criteria to act as a filter. Within this approach there are four important aspects: determining domain knowledge about the class of images, distinguishing the accept from the reject set, listing the criteria and constraints that serve as the basis of this general strategy, and developing alternative general strategies.

Under this general strategy there are several important questions that must be asked about a specific class of images for which the algorithm is being developed. These questions include the following,

- What is known about the objects of interest? (What stays the same, what varies?)

- What is known about the context of these objects?

- What is known about the kinds of noise processes which interfere with the image acquisition process?

- What is known about the sensor, its characteristics, anomalies, and ways of interacting with the subject?

The "application dependent" information obtained from these questions represents a priori constraints on the image class provided to the system by the user based on external knowledge about the imaging task. (e.g., if orientation is controlled or lighting is held constant). Additional constraints may be found by examining a broad range of examples and noting any major invariances seen.

There are several key aspects associated with distinguishing the Accept set from the Reject set. These aspects include,

- What stays the same, what varies across the image class, for desired (accept) vs undesired (reject) edges?

- Need for finding KEY facts to distinguish accept edges from reject ones. Alternatively, find a criteria subspace which consistently maximizes the difference between the accept and reject regions.

- As a general rule, we first try the smaller of the two sets in number, when looking for distinguishing characteristics. (Unless it just happens that the more numerous set has uniformly distinguishing characteristics which would allow it to be extracted more easily.)

- Look especially for measurement distributions (criteria dimensions) which completely or partially separate the two sets. Here the Max, and Min of each population is important. The Mode may also be useful (e.g., if mode grey value is 0). Average or Median is useful for judging rough skewness, or where the biggest "payoff" is. Often knowing the middle 80% of the distribution (10-90% rule) is of more value than Max/Min, which may represent outliers.

- Importance of using operations which MAINTAIN important DISTINCTIONS that exist between accept and reject set, while filtering out or eliminating other parts of the image.

- Need to consider what criteria/distinctions are affected by a given operation (i.e., added or eliminated).

- When choosing between alternative solutions, it is important to compare their relative costs. Costs of operations include: cost of testing for applicability and searching for the best parameters, time cost of the operation itself (how long it takes to execute), and expected cost of solving the sub-problems which it leaves. For a given expert, familiarity with operations will also play into the percieved cost-- i.e., there is an additional perceived cost of learning to how apply the operation correctly.

A fairly complete list of the types of criteria which may be used to distinguish accept edges from reject ones includes the following:

- Position (absolute/relative)

- Orientation (absolute/relative)

- Distance to something else

- Direction relative to something else

- Adjacency to, Connection to something else

- Continuity (broken vs continuous)

- Topology (closed loops vs open ended vs touching vs intersecting)

- Grey values (absolute levels)
- Colors (multispectral value clusters)
- Contrast (weak vs strong; as gradient vs as local peak or trough)
- Size (for lines, both a length dimension and a width dimension)
- Shape (straight vs curved; angles of intersection)
- Texture/Noise

Several alternative general strategies exist within the general strategy of separating desired from undesired edges. A partial listing of these strategies includes,

- Go for ALL of one of the sets (usually the smaller in number or the easier to extract), then eliminate parts of the other which still remain [SUP strategy].
- Go for ONLY one of the sets (the one which is easier to get a large subset of, such that NONE of the other remains), then add the parts which are still missing [INF strategy].
- Go for ONLY [INF] of both sets (accept and reject), follow this in parallel for both, use a competition tie-breaker to decide pixels which cannot be put in one or the other. Alternatively, try a parallel SUP strategy, using a tie-breaker to decide pixels which fall in both.
- Go for best split (least error) of the two sets, and then create subproblems to solve how to recover what is still missing and how to delete what should not be present.

4.3.2 General Approaches For Different Classes Of Lines

The type of lines that exist in images has a la,ge effect on what strategy to select when developing an algorithm. Among some of the general strategies to choose from are listed below organized by line type.

- Pulse Edges. Easier than step for single direction pulses, when background is generally smooth. Use opening/closing residue based on maximum pulse width.

- Step or Ramp Edges. Ease depends on width of edge vs background distortions. If edges are blurred or indistinct on a background with significant gradients, it may be difficult to separate the edges out. Use one of many types of edge generators (Morphological gradient, other gradients, Haralick operator, Marr operator, Rosenfeld multiresolution, etc.), then threshold absolutely or at multiple levels in combination with spatial information.

- Texture Edges. Try to convert to a step edge, then find the step edge as above. (Ex: Erode to eliminate one texture, Span/CDilate over to recover other side, dilate or close to fill in gaps, then find step. Alternatively, find both sides independently and have them compete for the middle). Skeleton of background ("Zone of influence") may help to modify the image in a way that would make it easier to find the texture boundaries. In general, the most important strategy for dealing with texture is to find a way to get rid of it.

- Step and Pulse. Trick here is to avoid getting steps on both sides of the pulse. Suggest a post-processing step which combines steps which are too close together.

- Uniform Texture and Step. Suppress texture as noise with complementary spatial filters; then go for step edge.

- Non-Uniform Texture and Step. Same method as above, or use dual-sided technique discussed under texture if there was a chance of inadvertently eliminating the step.

- Uniform Texture and Pulse. Skeletonize down, eliminate single points--long lines will stay. If lines are short, size may not discriminate, so may we need to distinguish between the shape differences of the texture vs the lines. For example, to discriminate thin objects from blobs--erosion gives min width dimension, skeleton gives max--if we perform both and the result is about the same, we have a roundish object, otherwise a more linear one.

- Pulse, Step and Uniform or Non-Uniform Texture. Remove texture, get steps while trying to avoid destroying pulses, cleanup afterward or join together things that are too close. To remove texture, consider shape or size approaches discussed above.

4.3.3 Suggested Methods For Specific Edge Types

Several different strategies for working with specific edge types have been developed over the years. A list of these methods is included below.

- Long, continuous, straight segments in different orientations. Use D. McCubbrey method (Union of erosions by shorter segments in several orientations; accept results of sufficient length) (Fails for curved or short or broken segments.)

- Generalization of above using correlation (shift-add instead of shift-and = erosion) can compensate for small breaks. Choose between these two based on presence of small breaks. (how small? how many?) Care needs to be taken in closing breaks, not to connect things which shouldn't be connected.

- Closed Loops (edges of closed figures). Skeletonizing including end point removal (pruning) will not eliminate such lines. (Fails if loops are in any way broken). Useful for separating closed figure edges from other types.

- Short Lines or Lines with Ends. Skeletonizing or thinning lines which reduces ends can lose them--the sooner the shorter. (Use non-pruning methods if you need to preserve them).

- Narrow Lines. (Pulse Lines) These can be removed by different filtering operations depending on how narrow they are. If the accept and reject sets differ in terms of line narrowness, then this fact can be exploited to select one or the other.

  Pulse Lines can be positive or negative in direction. There are symmetric (dual) operations for pulling out each type. (Opening, Closing residues). Likewise, noise comes in positive (salt) and negative (pepper) types, and there exist dual filters for taking out each type.

  Noise can play a complicating role for very narrow pulses. The operations which filter salt or pepper noise based on size will also wipe out pulse lines of the same directionality if they are not wider than the noise spots. (There are other techniques which can be used in this case, which make use of the fact the lines are extended, while the noise spots are usually compact).

- Intersecting or Touching Lines. These specific configurations can be easily found by the Cyto, so if your accept and reject sets of lines differ with respect to this, you can separate them by finding these configurations and using conditional dilation to recover the whole line. Given that the lines are intersecting or touching, we can ask about the angles of intersection (narrow, wide, right angles). This may help to separate the sets, or it may have a bearing on the choice of filters and parameters.

- Oriented Lines. (assumes Straight segments) If lines are curved, the whole question of orientation becomes irrelevant.

  For limited numbers of specific orientations, we can design special structuring elements (take union of openings) or rotate the image by a specific number of degrees. (For example, for orthogonal segments not oriented with x/y axes).

  For segments oriented with x/y axes, specific linear structuring elements are already available. (Matched Filter approaches) These are the cubes and walls (square and linear cross-sections). If one set of edges is oriented and the other is not, we can use these elements to separate them.

  Non-oriented lines suggest the use of rotationally invariant elements like circles or disks, or unions of all different oriented short segments (slower).

- Broken Lines. If noise exists, or if angles of intersecting lines are narrow, then attempts to fill in broken line gaps may create spurious lines between noise points, or fill in the corners of the narrow angles. This requires carefully choosing parameters based on the size of gaps, and the angle of intersections.

Under this general strategy there are two contrast issues worth noting:

- If the edges of interest differ in contrast from those which are not, we can use this to separate them. (Work with some form of gradient image). It is best that the desired edges have the most contrast.

- There is an interplay between the contrast, the width of the edge or ramp, and the noise, which limits the types of filtering which can be done.

4.4 System Design

This project has produced an overall system design for an image algorithmist aid for edge extraction. An overview of the design is shown in figure 4-5. Five major system components shown in this figure: the man/machine interface, the inference engine, the knowledge base, the system goals, and the current image characteristics.

System design specifications for the man/machine interface component have been developed in this project. These specifications were also partially implemented. The remainder will be implemented in the second phase when the decision aid is embedded into the image processing workstation. These specifications include a restricted natural language interface, an explanation capability, an elaboration facility, a help mechanism, and a definition lookup capability.

The inference engine component is fully implemented. It is described in great detail in the inference engine subsection of this document.

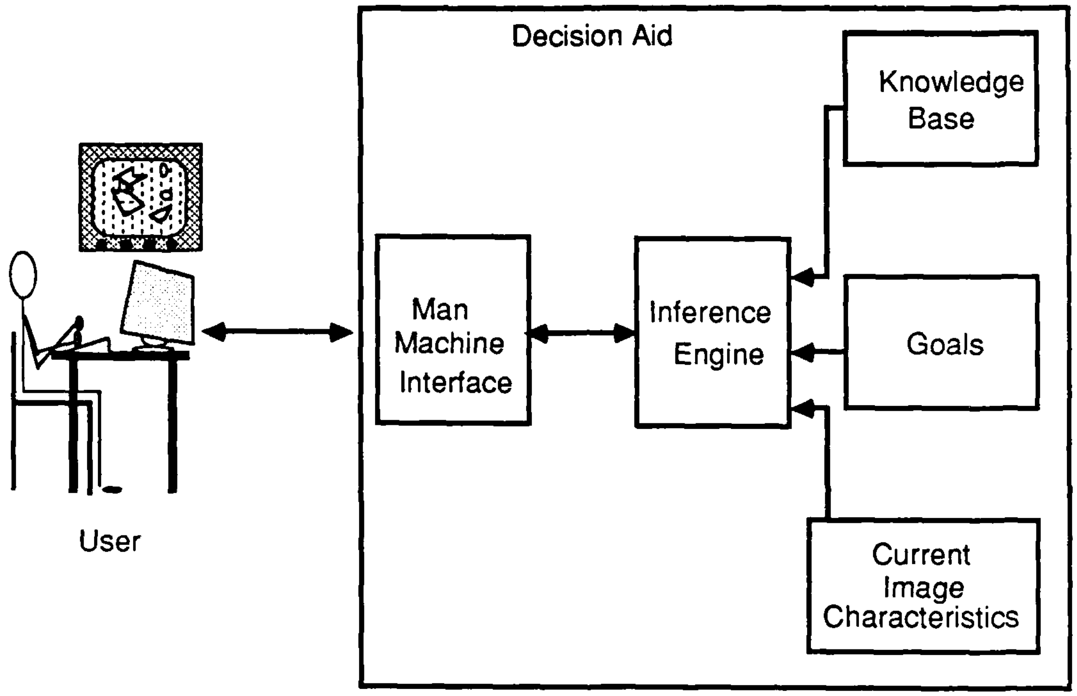
The knowledge base component was partially implemented under this project. Development of the knowledge base for the background/noise removal phase of the edge extraction algorithm sequence is nearly completed. The remaining development of this component is expected to continue through phase II of the program. The methodology that will be employed will be to individually develop knowledge structures for the phases of the edge extraction algorithm sequence outlined in figure 4-6. As illustrated in this figure, the sequence consists of the following phases.

- BACKGROUND/NOISE REMOVAL PHASE. Use of different kinds of filters (openings, closings, spike-fil, donutfil, etc.) to remove noise. If large undesired regions of the background can be identified, they can also be removed at this point.

- EDGE GENERATION PHASE. Pulses are recovered by opening residues (positive) or closing residues (negative). Edges are generated by applying one of myriad possiblities (morphological gradient, Haralick edge detectors, Gaussian-Laplacian, Sobel, Roberts, Rosenfeld multiresolution method, etc). All of these generate grey-level edge images.

- GREY-LEVEL ENHANCEMENT PHASE. All sorts of grey-level massaging techniques may be used here to improve the output of the edge generator before the threshold is applied. These include template matching (erosion by walls) or correlation, grey level smoothings or skeletons or thickenings. .le

AI-Based, Image Algorithm Decision Aid



System Overview

Figure 4-5

- BACKGROUND/NOISE REMOVAL PHASE
- EDGE GENERATION PHASE
- GREY-LEVEL ENHANCEMENT PHASE
- THRESHOLD GENERATION PHASE
- BINARY ENHANCEMENT PHASE
- DISCRIMINATION PHASE
- CLEANUP, POST-PROCESSING PHASE

Overview of Edge Extraction Algorithm Sequence

Figure 4-6

THRESHOLD GENERATION PHASE. The threshold can be "above" (to get the edges) or "below" (to get the flat regions). In both cases, a single simple threshold may be used, or a multiple-level threshold which starts first with the most certain regions, and adds increasingly uncertian pixels based on an additional spatial proximity criterion such as "within convex hull" or "within d distance" of previous result.

- BINARY ENHANCEMENT PHASE. A multitude of binary massaging techniques to improve the binary result of the threshold operation after the fact. These include skeleton or thinning operations to reduce the size of edges, openings to select edges which are long or straight or properly oriented, closings or Hough transforms or other operations which close up small gaps in broken edges, and so on. If we focus on regions, then closings which fill in small holes, homo-topic thinnings to grow out the regions without touching, or smoothings to give them less ragged edges,etc.

- DISCRIMINATION PHASE. Once we have a decent binary result, we can now try to discriminate the edges we want from those we don't, based on distinguishing criteria such as length, width, orientation, position, topology, shape, etc. In some cases, some portion of these discriminations may have been made earlier, in one of the grey-level phases or even in the thresholding phase.

- CLEANUP, POST-PROCESSING PHASE. If there is anything left to do such as joining edges which were too close together, or eliminating edges which were too small, or wiping out spurious edges or points resulting from the discrimination technique used, here's the last chance

The systems goals component is defined to be the C3PL commands augmented by library procedures. Initially these goals will include only the commands themselves, although future work could be directed at selecting command parameters.

The current image characterstics component will initially be provided by the user through system interaction. It is expected, however, that much of this will be eventually automated through conventional software procedures, and facilities have been implemented in the inference engine that allow the input of image information from such procedures.

4.5 Knowledge Acquisition Environment

This research has resulted in a fully implemented knowledge acquisition software system, VisEd. It is described in detail in the software systems section.

4.6 Inference Engine

This research has resulted in a fully implemented inference engine, ARC. It is described in detail in the software systems section.

4.7 Prototype Expert System

This research has produced a prototype expert system. This prototype incorporates knowledge about filtering commands to aid the user in selecting the appropriate command for processing an image. This prototype implements much of the first phase of the edge extraction sequence outlined above. An example of the dialog produced by this system follows.

```
(01) Are there undesired bright objects ? NO

(02) The BUMP-FILTER does not look promising

(03) The OPEN-FILTER does not look promising

(04) Are there undesired dark objects ? YES

(05) Are the dark elements you wish to remove
     narrow ? ELABORATION

(06) ie. They are generally more narrow than the desired
     dark objects in the image.

(07) Are the dark elements you wish to remove narrow ? 3

(08) Are there undesired dark lines in the image ? YES

(09) Are the undesirable dark lines narrow  ? YES

(10) Are there isolated dark noise points ? YES

(11) Are there undesirable dark clusters in the
     image ? YES

(12) Are the undesirable dark clusters small ? YES
```

(13) The CLOSE-FILTER looks promising. Do you want to try it ? YES

(14) CLOSE-FILTER processing would occur here

(15) Do you require further filtering ?YES

(16) Are there bright isolated noise points in the image ? NO

(17) The SPIKE-FILTER does not look promising

(18) Are there primarily dark isolated noise points ? NO

(19) The HOLE-FILTER does not look promising

(20) The DONUT-FILTER does not look promising

(21) Are there undesired compact dark objects in the image. ? YES

(22) Are the dark compact undesired objects isolated from other dark regions ? YES

(23) The POTHOLE-FILTER looks promising. Do you want to try it ? YES

(24) POTHOLE-FILTER processing would occur here

(25) Do you require further filtering ?NO

(26) Reasoning Completed!

Several interesting points can be seen in this example. In line 01 a general but powerful question is asked of the user. The response, no, immediately indicates that the BUMP-FILTER (02) and the OPEN-FILTER (03) need no further consideration. The system then focuses on removing dark objects (04). The positive response in (04) immediately produces a more focused question (05). In response to this question the user ask for more elaboration on the question. The system responds with a more detailed description of the question (06). The system then restates the question (07), and the user responds with a three. This response indicates most of the elements are narrow, but not all. The system then continues the dialog by asking more questions that are concerned with using the CLOSE-FILTER (08)-(12). From the answers to these questions it concludes that the filter looks promising, and asks the user if he wants to try it (13). The user reponds positively, and the system attaches a procedure to do the processing (14). (At present the only action produced by this procedure is to print (14), but in future versions actual processing will occur at this stage.) After the processing is is completed, the system asks the user if additional goal investigation is necessary (15). The user responds positively, and the system continues to investigate filters. No bright noise in the image (16) eliminates the SPIKE-FILTER from further consideration (17). No dark isolated noise points (18) eliminates the HOLE-FILTER (19) and the DONUT-FILTER (20) as possible filters. Positive undesired (21), isolated (22), dark compact objects make the POTHOLE-FILTER look promising (23), and it is processed (24). No further filtering is required (25), and the reasoning is completed (26).

5 CONCLUSION

This project was very successful. Several positive outcomes have been been produced through this research including a working prototype expert system that aids its user in selecting an appropriate image processing filtering operation. All of the project goals were obtained. In fact expectations for this project were greatly exceeded.

DATE
ILMED
8